目录

基础篇

一、JDK 常用的包

♦java.lang： 这个是系统的基础类，比如 String、Math、Integer、System 和 Thread，提供常用功能。在 java.lang 包中还有一个子包：java.lang.reflect 用于实现 java 类...
♦java.io: 这里面是所有输入输出有关的类，比如文件操作等
♦java.net: 这里面是与网络有关的类，比如 URL,URLConnection 等。
♦java.util：这个是系统辅助类，特别是集合类 Collection,List,Map 等。
♦java.sql: 这个是数据库操作的类，Connection, Statememt，ResultSet 等

二、Get 和 Post 的区别

1.get 是从服务器上获取数据，post 是向服务器传送数据，
2.get 传送的数据量较小，不能大于 2KB。post 传送的数据量较大，一般被默认为不受限制。
3.get 安全性非常低，post 安全性较高。但是执行效率却比 Post 方法好。
4.在进行文件上传时只能使用 post 而不能是 get。

三、Java 多态的具体体现

面向对象编程有四个特征：抽象，封装，继承，多态。
多态有四种体现形式：

1. 接口和接口的继承。
2. 类和类的继承。

3. 重载。
4. 重写。

其中重载和重写为核心。

重载：重载发生在同一个类中，在该类中如果存在多个同名方法，但是方法的参数类型和个数不一样，那么说明该方法被重载了。

重写：重写发生在子类继承父类的关系中，父类中的方法被子类继承，方法名，返回值类型，参数完全一样，但是方法体不一样，那么说明父类中的该方法被子类重写了。

四、StringBuffer StringBuilder String 区别

String	字符串常量	不可变	使用字符串拼接时是不同的 2 个空间
StringBuffer	字符串变量	可变	线程安全 字符串拼接直接在字符串后追加
StringBuilder	字符串变量	可变	非线程安全 字符串拼接直接在字符串后追加

1.StringBuilder 执行效率高于 StringBuffer 高于 String.
2.String 是一个常量，是不可变的，所以对于每一次+=赋值都会创建一个新的对象，StringBuffer 和 StringBuilder 都是可变的，当进行字符串拼接时采用 append 方法，在原来的基础上进行追加，所以性能比 String 要高，又因为 StringBuffer 是线程安全的而 StringBuilder 是线程非安全的，所以 StringBuilder 的效率高于 StringBuffer.
3.对于大数据量的字符串的拼接，采用 StringBuffer,StringBuilder.

五、Hashtable 与 HashMap 的区别

HashMap 不是线程安全的，HashTable 是线程安全。
HashMap 允许空（null）的键和值（key），HashTable 则不允许。
HashMap 性能优于 Hashtable。

Map
1.Map 是一个以键值对存储的接口。Map 下有两个具体的实现，分别是 HashMap 和 HashTable.
2.HashMap 是线程非安全的，HashTable 是线程安全的，所以 HashMap 的效率高于 HashTable.
3.HashMap 允许键或值为空，而 HashTable 不允许键或值为空.

六、九大隐式对象

输入/输出对象： request response out
作用域通信对象： session application pageContext

Servlet 对象： page config
错误对象： exception

七、Forword(请求转发)与 Redirect(重定向)

1、从数据共享上
Forword 是一个请求的延续，可以共享 request 的数据
Redirect 开启一个新的请求，不可以共享 request 的数据
2、从地址栏
Forword 转发地址栏不发生变化
Redirect 转发地址栏发生变化

八、JQurey 总结

jquery 是一个轻量级的 js 框架，具有跨浏览器的特性，兼容性好，
并且封装了很多工具，方便使用。
常用的有: 选择器 ，dom 操作 ，ajax(ajax 不能跨域) ，特效，工具类

九、XML 和 Json 的特点

Xml 特点：

1、有且只有一个根节点；

2、数据传输的载体

3、所有的标签都需要自定义

4、是纯文本文件

Json（JavaScript Object Notation）特点：

json 分为两种格式:

json 对象(就是在{}中存储键值对，键和值之间用冒号分隔，

键 值 对之间用逗号分隔);

json 数组(就是[]中存储多个 json 对象，json 对象之间用逗号分隔)

（两者间可以进行相互嵌套）数据传输的载体之一

区别：

传输同样格式的数据，xml 需要使用更多的字符进行描述，

流行的是基于 json 的数据传输。

xml 的层次结构比 json 更清晰。

共同点：

xml 和 json 都是数据传输的载体，并且具有跨平台跨语言的特性。

十、request.getSession()、reqeust.getSession(false)和 request.getSession(true)

getSession()/getSession(true)：当 session 存在时返回该 session，否则新建一个 session 并返回该对象
getSession(false)：当 session 存在时返回该 session，否则返回 null

十一、Page 和 PageContext 的区别

Page 是 servlet 对象；使用 this 关键字，它的作用范围是在同一页面。
PageContext 是作用域通信对象；通常使用 setAttribute()和 getAttribute()来设置和获取存放对象的值。

十二、Ajax 总结

AJAX 全称： 异步 JavaScript 及 XML（Asynchronous JavaScript And XML）
Ajax 的核心是 JavaScript 对象 XmlHttpRequest(XHR)。

Ajax 的优点:
提高用户体验度(UE)
提高应用程序的性能
进行局部刷新
AJAX 不是一种新的编程语言，而是一种用于创建更好更快以及交互性更强的 Web 应用程序的技术。
2. 通过 AJAX，我们的 JavaScript 可使用 JavaScript 的 XMLHttpRequest 对象来直接与服务器进行通信。通过这个对象，我们的 JavaScript 可在不重载页面的情况与 Web 服务器交换数据，即可局部刷新。

3. AJAX 在浏览器与 Web 服务器之间使用异步数据传输（HTTP 请求），这样就可使网页从服务器请求少量的信息，而不是整个页面，减轻服务器的负担，提升站点的性能。
AJAX 可使因特网应用程序更小、更快，更友好，用户体验（UE）好。
5. Ajax 是基于标准化并被广泛支持的技术，并且不需要插件和下载小程序

十三、JSP9 大隐视对象中四个作用域的大小与作用范围

四个作用域从大到小：appliaction>session>request>page
application：全局作用范围，整个应用程序共享.生命周期为：应用程序启动到停止。
session：会话作用域，当用户首次访问时，产生一个新的会话，以后服务器就可以记住这个会话状态。
request：请求作用域，就是客户端的一次请求。
page：一个 JSP 页面。
以上作用范围使越来越小， request 和 page 的生命周期都是短暂的，他们之间的区别就是：一个 request 可以包含多个 page 页(include，forward)。

十四、List,Set,Collection,Collections

1.List 和 Set 都是接口，他们都继承于接口 Collection,List 是一个有序的可重复的集合，而 Set 的无序的不可重复的集合。Collection 是集合的顶层接口，Collections 是一个封装了众多关于集合操作的静态方法的工具类,因为构造方法是私有的，所以不能实例化。

2.List 接口实现类有 ArrayList,LinkedList,Vector。ArrayList 和 Vector 是基于数组实现的,所以查询的时候速度快，而在进行增加和删除的时候速度较慢 LinkedList 是基于链式存储结构 ,所以在进行查询的时候速度较慢但在进行增加和删除的时候速度较快。又因为 Vector 是线程安全的，所以他和 ArrayList 相比而言，查询效率要低。

十五、java 的基本数据类型

数据类型 大小
byte(字节) 1(8 位)
shot(短整型) 2(16 位)
int(整型)4(32 位)
long(长整型) 8(32 位)
float(浮点型) 4(32 位)
double(双精度) 8(64 位)
char(字符型) 2(16 位)

boolean(布尔型) 1 位

十六、冒泡排序

```
public class Sort {
    public static void sort() {
        Scanner input = new Scanner(System.in);
        int sort[] = new int[10];
        int temp;
        System.out.println("请输入 10 个排序的数据：");
        for (int i = 0; i < sort.length; i++) {
            sort[i] = input.nextInt();
        }
        for (int i = 0; i < sort.length - 1; i++) {
            for (int j = 0; j < sort.length - i - 1; j++)                {
                if (sort[j] < sort[j + 1]) {
                    temp = sort[j];
                    sort[j] = sort[j + 1];
                    sort[j + 1] = temp;
                }
            }
        }
        System.out.println("排列后的顺序为：");
        for(int i=0;i<sort.length;i++){
            System.out.print(sort[i]+"    ");
        }
    }
    public static void main(String[] args) {
        sort();
    }
}
```

十七、二分查找法

手写冒泡排序

```
int[] arr = new int[]{43,32,76,-98,0,64,33,-21,32,99};

    //冒泡排序
    for(int i = 0;i < arr.length - 1;i++){

        for(int j = 0;j < arr.length - 1 - i;j++){

            if(arr[j] > arr[j + 1]){
```

```
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }

        }
```

```
// 如果从大型的数组中查询数据,可以使用二分法,也称为折半法
// 注意
// 1.使用二分法的时候,只能针对排序好的数组
// 2.数组中不能用重复的数据.
class Test02_Array2{
    public static void main(String [] args){
        int [] arr = {10,11,12,13,14,15,16,17,18,19};
        int key = 20; // 需要从数组中查询的数据

        int start = 0; // 定义一个变量指向数组的 0 号索引
        int end = arr.length -1; // 定义一个变量,指向数组的最后一个元素

        boolean flag = false;
        int middle = -1;
        while(start <= end){
            middle = (start + end) / 2;
            if(arr[middle] == key){
                flag = true;
                break;
            }
            // 例如,我们需要中 18,但是 arr[middle]是 14
            else if(arr[middle] < key){
                start = middle +1;
            }
            // 我们需要中 18,但是 arr[middle]是 19
            else{
                end = middle -1;
            }
        }

        if(flag){
            System.out.println("找到数据,索引是:" + middle);
        }else{
            System.out.println("没有找到数据");
        }
    }}
```

十八、时间类型转换

```
public class DateFormat {
    public static void fun() {
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy 年 MM 月 dd 日");
        String newDate;
        try {
            newDate = sdf.format(new SimpleDateFormat("yyyyMMdd")
                    .parse("20121115"));
            System.out.println(newDate);
        } catch (ParseException e) {
            e.printStackTrace();
        }
    }
    public static void main(String args[]) {
        fun();
    }
}
```

十九、阶乘

```
public class Multiply {
    public static int multiply(int num) {
        if (num < 0) {
            System.out.println("请输入大于 0 的数！");
            return -1;
        } else if (num == 0 || num == 1) {
            return 1;
        } else {
            return multiply(num - 1) * num;
        }
    }
    public static void main(String[] args) {
        System.out.println(multiply(10));
    }
}
```

二十、UE 和 UI 的区别

UE 是用户体验度
UI 界面原型（用户界面）(相当于买房时用的模型）

设计 UI 的作用：
1、帮助程序员工作（界面已由美工设计完成）
2、提前让用户对项目有个宏观的了解，知道效果是什么样子。

二十一、osi 七层模型

第一层：物理层
第二层：数据链路层
第三层：网络层
第四层：传输层
第五层：会话层
第六层：表示层
第七层：应用层

二十二、线程和进程的区别

1.线程(Thread)与进程（Process）
进程定义的是应用程序与应用程序之间的边界，通常来说一个进程就代表一个与之对应的应用程序。不同的进程之间不能共享代码和数据空间，而同一进程的不同线程可以共享代码和数据空间。

2.一个进程可以包括若干个线程，同时创建多个线程来完成某项任务，便是多线程。

3.实现线程的两种方式：继承 Thread 类，实现 Runable 接口

二十三、jvm 的内存结构

java 虚拟机的内存结构分为堆(heap)和栈(stack),堆里面存放是对象实例也就是 new 出来的对象。栈里面存放的是基本数据类型以及引用数据类型的地址。

对于所谓的常量是存储在方法区的常量池里面。

二十四、内存泄露和内存溢出

内存泄露（memory leak)，是指应用程序在申请内存后，
无法释放已经申请的内存空间.一次内存泄露危害可以忽略，
但如果任其发展最终会导致内存溢出(out of memory).
如读取文件后流要进行及时的关闭以及对数据库连接的释放。

内存溢出（out of memory）是指应用程序在申请内存时，
没有足够的内存空间供其使用。
如我们在项目中对于大批量数据的导入，采用分段批量提交的方式。

二十五、单例

单例就是该类只能返回一个实例。

单例所具备的特点：
1.私有化的构造函数
2.私有的静态的全局变量
3.公有的静态的方法

单例分为懒汉式、饿汉式和双层锁式
饿汉式:

```
public class Singleton1 {
        private Singleton1() {};
        private static Singleton1 single = new Singleton1();
        public static Singleton1 getInstance() {
              return single;
      }
 }
```

懒汉式：

```
public class Singleton2 {
        private Singleton2() {}
        private static Singleton2 single=null;
        public tatic Singleton2 getInstance() {
               if (single == null) {
                    single = new Singleton2();
              }
             return single;
      }
 }
```

线程安全：

```
public class Singleton3 {
        private Singleton3() {}
        private static Singleton3 single ;
```

```
    public static Singleton3 getInstance() {
        if(null == single){
            synchronized(Singleton3.class){
                if(null == single){
                    single = new Singleton3();
                }
            }
        }
        return single;
    }
}
```

二十六、解析 xml 文件的几种技术

1、 解析 xml 的几种技术

1.dom4j

2.sax

3.jaxb

4.jdom

5.dom

1.dom4j

dom4j 是一个 Java 的 XML API ,类似于 jdom ,用来读写 XML 文件的。dom4j 是一个非常优秀的 Java XML API，具有性能优异、功能强大和极端易用使用的特点，同时它也是一个开放源代码的软件。

2.sax

SAX（simple API for XML）是一种 XML 解析的替代方法。相比于 DOM，SAX 是一种速度更快，更有效的方法。它逐行扫描文档，一边扫描一边解析。而且相比于 DOM，SAX 可以在解析文档的任意时刻停止解析，但任何事物都有其相反的一面，对于 SAX 来说就是操作复杂。

3.jaxb

JAXB（Java Architecture for XML Binding）是一个业界的标准，是一项可以根据 XML Schema 产生 Java 类的技术。该过程中，JAXB 也提供了将 XML 实例文档反向生成 Java 对象树的方法，并能将 Java 对象树的内容重新写到 XML 实例文档。从另一方面来讲，JAXB 提供了快速而简便的方法将 XML 模式绑定到 Java 表示，从而使得 Java 开发者在 Java 应用程序中能方便地结合 XML 数据和处理函数。

2、dom4j 与 sax 之间的对比：【注：必须掌握 !】

dom4j 不适合大文件的解析，因为它是一下子将文件加载到内存中，所以有可能出现内存溢出，

sax 是基于事件来对 xml 进行解析的，所以他可以解析大文件的 xml

也正是因为如此，所以 dom4j 可以对 xml 进行灵活的增删改查和导航，而 sax 没有这么强的灵活性

所以 sax 经常是用来解析大型 xml 文件，而要对 xml 文件进行一些灵活（crud）操作就用 dom4j

二十七、项目的生命周期

1.需求分析
2.概要设计
3.详细设计(用例图，流程图，类图)
4.数据库设计(powerdesigner)
5.代码开发（编写）
6.单元测试（junit 白盒测试）(开发人员)
svn 版本管理工具(提交，更新代码，文档)
7.集成测试 （黑盒测试，loadrunner（编写测试脚本）(高级测试)）
8.上线试运行 （用户自己体验）
9.压力测试（loadrunner）
10.正式上线
11.维护

二十八、OSCache 的判断

```
Object obj = CacheManager.getInstance().getObj("oaBrandList");
//从缓存中取数据
if (null == obj)  {
        obj = brandDao.getBrandList();
        //如果为空再从数据库获取数据
         //获取之后放入缓存中
        CacheManager.getInstance().putObj("oaBrandList", obj);
    }
    return (List<OaBrand>)obj;
```

二十九、经常访问的技术网站

1.csdn(详细步骤的描述)
2.iteye(详细步骤的描述)
3.oschina（开源中国获取 java 开源方面的信息技术）
4.java 开源大全 www.open-open.com(获取 java 开源方面的信息技术)
5.infoq(对 java,php,.net 等这些语言的一些最新消息的报道)
6.itpub 综合性技术论坛

三十、项目团队中交流的工具

飞秋(局域网)　　qq(局域网，外网)

RTX（局域网，外网）邮箱 （局域网，外网）

三十一、平时浏览的书籍

实战经验：
*** in action(实战)
*** 深入浅出
*** 入门指南
思想基础:
大话设计模式　　重构

三十二、java Exception 体系结构

java 异常是程序运行过程中出现的错误。Java 把异常当作对象来处理，并定义一个基类 java.lang.Throwable 作为所有异常的超类。在 Java API 中定义了许多异常类,分为两大类，错误 Error 和异常 Exception。其中异常类 Exception 又分为运行时异常(RuntimeException)和非运行时异常(非 runtimeException)，也称之为不检查异常（Unchecked Exception）和检查异常（Checked Exception）。

1、Error 与 Exception
Error 是程序无法处理的错误，比如 OutOfMemoryError、ThreadDeath 等。
这些异常发生时，Java 虚拟机（JVM）一般会选择线程终止。

Exception 是程序本身可以处理的异常，这种异常分两大类运行时异常和非运行时异常。程序中应当尽可能去处理这些异常。

2、运行时异常和非运行时异常

运行时异常: 都是 RuntimeException 类及其子类异常：

IndexOutOfBoundsException 索引越界异常

ArithmeticException：数学计算异常

NullPointerException：空指针异常

ArrayOutOfBoundsException：数组索引越界异常

ClassNotFoundException：类文件未找到异常

ClassCastException：造型异常（类型转换异常）

这些异常是不检查异常（Unchecked Exception），程序中可以选择捕获处理，也可以不处理。这些异常一般是由程序逻辑错误引起的。

非运行时异常:是 RuntimeException 以外的异常，类型上都属于 Exception 类及其子类。从程序语法角度讲是必须进行处理的异常，如果不处理，程序就不能编译通过。如：

IOException、文件读写异常

FileNotFoundException：文件未找到异常

EOFException：读写文件尾异常

MalformedURLException：URL 格式错误异常

SocketException：Socket 异常

SQLException：SQL 数据库异常

三十三、session 和 cookie 的区别

session 是存储在服务器端，cookie 是存储在客户端的，所以安全来讲 session 的安全性要比 cookie 高 ,然后我们获取 session 里的信息是通过存放在会话 cookie 里的 sessionid 获取的。又由于 session 是存放在服务器的内存中，所以 session 里的东西不断增加会造成服务器的负担，所以会把很重要的信息存储在 session 中，而把一些次要东西存储在客户端的 cookie 里，然后 cookie 确切的说分为两大类分为会话 cookie 和持久化 cookie，会话 cookie 确切的说是存放在客户端浏览器的内存中 ,所以说他的生命周期和浏览器是一致的，浏览器关了会话 cookie 也就消失了，然而持久化 cookie 是存放在客户端硬盘中，而持久化 cookie 的生命周期就是我们在设置 cookie 时候设置的那个保存时间，然后我们考虑一问题当浏览器关闭时 session 会不会丢失 ,从上面叙述分析 session 的信息是通过 sessionid 获取的，而 sessionid 是存放在会话 cookie 当中的，当浏览器关闭的时候会话 cookie 消

失所以我们的 sessionid 也就消失了，但是 session 的信息还存在服务器端，这时我们只是查不到所谓的 session 但它并不是不存在。那么，session 在什么情况下丢失，就是在服务器关闭的时候，或者是 sessio 过期，再或者调用了 invalidate()的或者是我们想要 session 中的某一条数据消失调用 session.removeAttribute()方法，然后 session 在什么时候被创建呢，确切的说是通过调用 session.getsession 来创建，这就是 session 与 cookie 的区别

三十四、字节流与字符流的区别

stream 结尾都是字节流，reader 和 writer 结尾都是字符流

两者的区别就是读写的时候一个是按字节读写，一个是按字符。

实际使用通常差不多。

在读写文件需要对内容按行处理，比如比较特定字符，处理某一行数据的时候一般会选择字符流。

只是读写文件，和文件内容无关的，一般选择字节流。

三十五、final,finally,finalize 三者区别

Final 是一个修饰符：

当 final 修饰一个变量的时候，变量变成一个常量，它不能被二次赋值

当 final 修饰的变量为静态变量（即由 static 修饰）时，必须在声明这个变　量的时候给它赋值

当 final 修饰方法时，该方法不能被重写

当 final 修饰类时，该类不能被继承

Final 不能修饰抽象类，因为抽象类中会有需要子类实现的抽 象方法，(抽 象类中可以有抽象方法，也可以有普通方法，当一个抽象类中没有抽象方 法时，这个抽象类也就没有了它存在的必要）

Final 不能修饰接口，因为接口中有需要其实现类来实现的方法

Finally：

Finally 只能与 try/catch 语句结合使用，finally 语句块中的语句一定会执行， 并且会在 return，continue，break 关键字之前执行

finalize：

Finalize 是一个方法，属于 java.lang.Object 类，finalize()方法是 GC （ garbage collector 垃圾回收）运行机制的一部分，finalize()方法是在 GC 清理它所从 属的对象时被调用的

三十六、Io流的层次结构

从流的方向
输入流 输出流

从流的类型上
字符流 字节流

inputstream 和 outputstream 都是抽象类

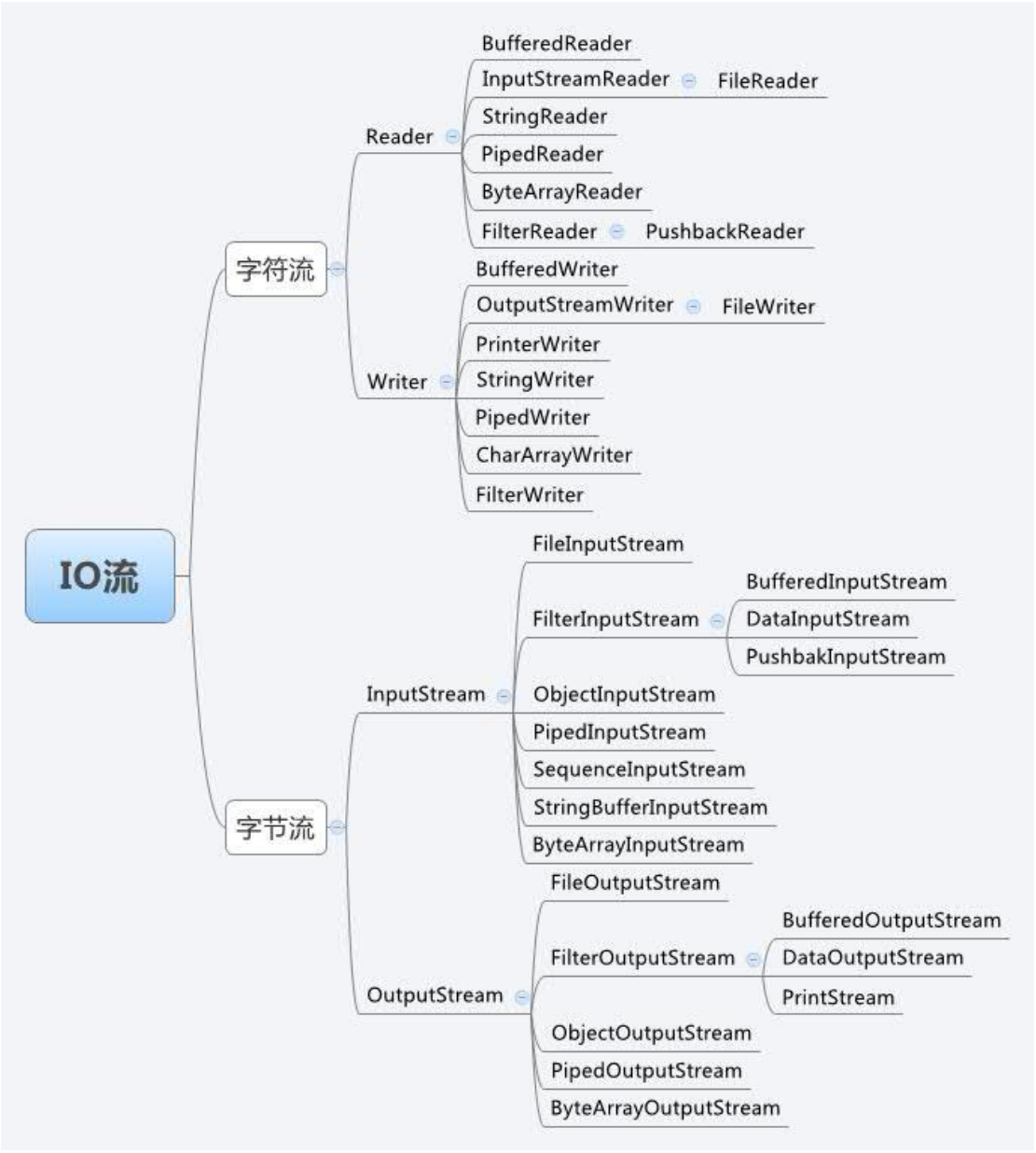
它们下面的实现包括

FileInputStream,BufferedInputStream

FileOutputStream,BufferedOutputStream

reader 和 writer

FileReader,BufferedReader,StringReader
FileWriter,BufferedWriter,StringWriter,PrintWriter



三十七、JAVA:

Java 是面向对象的，跨平台的，它通过 java 虚拟机来进行跨平台操作，它可以进行自动垃圾回收的【c 语言是通过人工进行垃圾回收】，java 还会进行自动分配内存。【c 语言是通过指定进行分配内存的】，只需要 new 一个对象，这个对象占用了多少空间，不需要我们来管，java 虚拟机负责管这些，用完之后也不需要我们来释放，java 虚拟机会自动释放

三十八、JavaSE JavaEE JavaME 区别

是什么：
Java SE=Java Standard Edition=j2se = java 标准版
Java EE=Java Enterprise Edition=j2ee= java 企业版
Java ME=Java Mobile Edition=j2me = java 移动版
特点：
SE 主要用于桌面程序（swing）,控制台开发(main 程序)。

EE 企业级开发(JSP,EJB,Spring MVC,Struts,hibernate,ibatis 等)，
用于企业级软件开发，网络开发，web 开发。

ME 嵌入式开发(手机,小家电，PDA)。[苹果的 ios，黑莓]

三者之间的关系：
Java SE（Java Platform, Standard Edition，Java 标准版）就是基于 JDK 和 JRE 的。

Java SE 为 Java EE 提供了基础。

Java EE 除了基于我们这个所谓的 Java SE 外，还新加了企业应用所需的类库

三十九、JDK　JRE　JVM 的区别：

Jdk【Java Development ToolKit】就是 java 开发工具箱， JDK 是整个 JAVA 的核心里边包含了 jre，它除了包含 jre 之外还包含了一些 javac 的工具类，把 java 源文件编译成 class 文件，java 文件是用来运行这个程序的，除此之外，里边还包含了 java 源生的 API，java.lang.integer 在 rt 的 jar 包里边【可以在项目中看到】，通过 rt 这个 jar 包来调用我们的这些 io 流写入写出等

JDK 有以下三种版本：

J2SE，standard edition，标准版，是我们通常用的一个版本

J2EE，enterpsise edtion，企业版，使用这种 JDK 开发 J2EE 应用程序

J2ME，micro edtion，主要用于移动设备、嵌入式设备上的 java 应用程序

Jre【Java　Runtime　Enviromental】是 java 运行时环境，那么所谓的 java 运行时环境，就是为了保证 java 程序能够运行时，所必备的一基础环境，也就是它只是

保证 java 程序运行的，不能用来开发，而 jdk 才是用来开发的，所有的 Java 程序都要在 JRE 下才能运行。

包括 JVM 和 JAVA 核心类库和支持文件。与 JDK 相比，它不包含开发工具——编译器、调试器和其它工具。

Jre 里边包含 jvm

Jvm：【Java Virtual Mechinal】因为 jre 是 java 运行时环境，java 运行靠什么运行，而底层就是依赖于 jvm，即 java 虚拟机，java 虚拟机用来加载类文件，java 中之所以有跨平台的作用，就是因为我们的 jvm

关系：

J2se 是基于 jdk 和 jre，

JDK 是整个 JAVA 的核心里边包含了 jre，

Jre 里边包含 jvm

四十、报错的状态码：

301 永久重定向

302 临时重定向

304 服务端 未改变

403 访问无权限

200 正常

404 路径

500 内部错误

四十一、协议以及默认的端口号

ftp 21 文件传输协议

Pop3 110　它是因特网 <http://baike.baidu.com/view/1706.htm>电子邮件 <http://baike.baidu.com/view/1524.htm> 的第一个离线 <http://baike.baidu.com/view/113466.htm>协议标准

Smtp　25　简单邮件传输协议

http　80　超文本传输协议

oracle 默认端口号 1521

mysql 默认端口号 3306

四十二、抽象类与接口的区别

1.一个类只能进行单继承，但可以实现多个接口。

2.有抽象方法的类一定是抽象类，但是抽象类里面不一定有抽象方法；
接口里面所有的方法的默认修饰符为 public abstract，接口里的成员变量默认的修饰符为 pulbic static final。

关系	
接口和接口	继承
接口和抽象类	抽象类实现接口
类和抽象类	类继承抽象类
类和类	继承

四十三、修饰符的作用

修饰符的作用范围:

	private	default	protected	public
同一个类中	可以	可以	可以	可以
同一个包的类中		可以	可以	可以
不同包的子类中			可以	可以
不同包的类中				可以

四十四、onready 和 onload 的区别

1.onready 比 onload 先执行

2.onready 是在页面解析完成之后执行，而 onload 是在页面所有元素加载后执行

3.onload 只执行最后一个而 onready 可以执行多个。

参考：

1.执行时间 window.onload 必须等到页面内包括图片的所有元素加载完毕后才能执行。 $(document).ready()是 DOM 结构绘制完毕后就执行，不必等到加载完毕。 2.编写个数不同 window.onload 不能同时编写多个，如果有多个 window.onload 方法，只会执行一个 $(document).ready()可以同时编写多个，并且都可以得到执行 3.简化写法 window.onload 没有简化写法 $(document).ready(function(){}) 可以简写成 $(function(){});

以浏览器装载文档为例，在页面加载完毕后，浏览器会通过 Javascript 为 DOM 元素添加事件。在常规的 Javascript 代码中，通常使用 window.onload 方法，而在 Jquery 中，使用的是 $(document).ready() 方法。 $(document).ready() 方法是事件模块中最重要一个函数，可以极大的提高 Web 应用程序的速度。

	window.load	$(document).ready()
执行时机	必须等待网页中所有的内容加载完毕后（包括图片）才能执行	网页中所有 DOM 结构绘制完毕后就执行，可以能 DOM 元素关联的内容并没有加载完

编写个数	不能同时编写多个
	以下代码无法正确执行：
	window.onload = function(){
	alert("text1");
	};
	window.onload = function(){
	alert("text2");
	};
	结果只输出第二个
		能同时编写多个
		以下代码正确执行：
		$(document).ready(function(){
		alert("Hello World");
		});
		$(document).ready(function(){
		alert("Hello again");
		});
		结果两次都输出

简化写法	无	$(function(){
		// do something
		});

另外，需要注意一点，由于在$(document).ready() 方法内注册的事件，只要 DOM 就绪就会被执行，因此可能此时元素的关联文件未下载完。例如与图片有

关的 html 下载完毕，并且已经解析为 DOM 树了，但很有可能图片还没有加载完毕，所以例如图片的高度和宽度这样的属性此时不一定有效。要解决这个问题，可以使用 Jquery 中另一个关于页面加载的方法---load() 方法。Load() 方法会在元素的 onload 事件中绑定一个处理函数。如果处理函数绑定给 window 对象，则会在所有内容(包括窗口、框架、对象和图像等) 加载完毕后触发，如果处理函数绑定在元素上，则会在元素的内容加载完毕后触发。 Jquery 代码如下： $(window).load(function (){ // 编写代码 });等价于 JavaScript 中的以下代码 Window.onload = function (){ // 编写代码 }

数据库篇

一、JDBC 连接数据库步骤(以 MYSQL 为例)

1、加载 JDBC 驱动程序：
通过 Class 类的 forName 方法实现，并将驱动地址放进去
成功加载后，会将 Driver 类的实例注册到 DriverManager 类中。
2、提供 JDBC 连接的 URL 、创建数据库的连接
•要连接数据库，需要向 java.sql.DriverManager 请求并获得 Connection 对象，该对象就代表一个数据库的连接。
•使用 DriverManager 的 getConnectin()方法传入指定的欲连接的数据库的路径、数据库的用户名和密码。
Connection con=DriverManager.getConnection(url , username , password);
&&&:"jdbc:mysql://localhost/test?user=root&password=123&useUnicode=true&characterEncoding=utf-8”；
3、创建一个 Statement
•要执行 SQL 语句，必须获得 java.sql.Statement 实例
•执行静态 SQL 语句。通常通过 Statement 实例实现。
•执行动态 SQL 语句。通常通过 PreparedStatement 实例实现。
String sql = “”；
Statement st = con.createStatement() ;
PreparedStatement pst = con.prepareStatement(sql) ;
4、执行 SQL 语句
Statement 接口提供了 executeQuery、executeUpdate、execute 三种方法
executeQuery：执行 select 语句，返回 ResultSet 结果集

ResultSet rst = pst.executeQuery();

- executeUpdate：执行 insert、update、delete 语句

pst.executeUpdate();

5、关闭 JDBC 对象

操作完成以后要把所有使用的 JDBC 对象全都关闭，以释放 JDBC 资源。

二、数据库连接池

数据库连接池的优点运行原理:

在我们不使用数据库连接池的时候，每次访问数据库都需要创建连接，

使用完成之后需要释放关闭连接，而这样是很耗费资源的。当我们使用

数据库连接池的时候，在 tomcat 启动的时候就创建了指定数量的连接，

之后当我们程序使用的时候就直接从连接池里面取，而不需要创建，同理，

当我们使用完的时候也不需要关闭连接，而是将连接返回到连接池中，供

其他请求继续使用。

DBCP：比较稳定。

C3P0: 性能比较高。

三、mysql 的数据库导入导出

配置：

首先找到 mysql 的安装目录，进入 bin 目录下复制路径

将 mysql 的 bin 目录粘贴在计算机环境变量的 path 中

授权：

登录 mysql

将某张表的某个权限赋给某个用户

grant [select,insert,update,delete,create,drop] on [databaseName].[tableName] to [userName]@[userIP] identified by [‘连接口令’]

grant select,insert,update,delete,create,drop on oa_ssh.user to root@[IP] identified by 'root';

将所有库的所有权限赋给某个用户

grant all privileges on *.* to [userName]@[userIp] identified by [‘连接口令’]

grant all privileges on *.* to root@[IP] identified by ‘root';
将所有库的所有权限赋给所有用户
grant all privileges on *.* to root@'%' identified by ‘root’;

导出本地数据库：
mysqldump -u 用户名 -p 数据库名 > 磁盘：导出的文件名(加后缀)
远程导出数据库：
mysqldump -h IP -u 用户名 -p 数据库名称 >导出的文件名(加后缀)
远程导出数据表：
mysqldump -u root -p -d --add-drop-table 数据库名称 > 导出文件名(加后缀)

导入数据：
mysql -u root -p 登录成功后 ==》 source 磁盘：导入的文件名(加后缀)

四、jdbc 分段批量提交的时候出现异常怎么处理?

通过 Map 来解决性能问题。首先在分段批量提交的时候，我们不采用事务，这样就保证了合法的数据就自动提交，不合法的数据就自己自动进行回滚，为了避免不合法数据影响后续合法数据的提交，采用定义业务规则字典表，实现对数据的验证，将不合法的数据记录下来，供用户进行后续处理，而合法的数据就全部提交。

五、jdbc 批量处理数据

批量处理数据:(代码优化:提高程序执行性能)

降低了 java 程序代码(客户端)和数据库之间的 网络通信的次数。

在 jdbc 中进行批量插入的核心 API 为 addBatch,executeBatch

大数据量的插入问题:（jdbc,hibernate,ibatis）

1.每次只插入一条和数据库交互多次(很耗时间)

2.批量插入和数据库只交互一次(内存溢出)

3.分段批量插入(推荐)

jdbc 批量处理数据是通过 PreparedStatement 对象的 addbatch(), executebatch( ) clearbatch()进行和数据库的交互。通常我们使用分段批量处理的方式 这样可以提高程序的性能 ，防止内存溢出。

1.每个 sql 语句都和数据库交互一次(非批量操作)

2.只和数据库交互一次(批量操作)(内存溢出)

当数据达到一定额度的时候就和数据库进行交互，分多次进行(分段批量操作)

(500 或者 1000)

```
pst.addBatch();
    if (i > 0 && i%1000 == 0) {
                                pst.executeBatch();
                                pst.clearBatch();
    }
```

六、Oracle 分页

select * from (select * from (select s.*,rownum rn from student s) where rn<=5) where rn>0

七、Oracle 的基本数据类型

Oracle 的基本数据类型（常用）:

1、字符型

Char 固定长度字符串 占 2000 个字节

Varchar2 可变长度字符串 占 4000 个字节

Nvarchar2 占 2000 个字符（最多能存 2000 个字母/中文）

2、大对象型（lob）

Blob ：二进制数据 最大长度 4G

Blob 用于存一些图片，视频，文件。

比如：当我们在进行文件上传时，我们一般把上传的文件存在硬盘上，可以不占用数据库，下载时，如果项目迁移时，文件也要跟着迁移。因此我们可以把用 blob 把它存在数据库中。但这样也增加了数据库的负担。

Clob ：字符数据 最大长度 4G，可以存大字符串 varchar2 和 nvarchar2 都具有一定的局限性，它们长度有限，但数据库中无论用 varchar2 或 nvarchar2 类型，还是用 clob，在 java 端都使用 String 接收。

3、数值型

Integer 整数类型，小的整数。

Float 浮点数类型。

Real 实数类型。
Number（p,s）包含小数位的数值类型。P 表示精度，s 表示小数后的位数。
Eg: number(10,2) 表示小数点之前可有 8 位数字，小数点后有 2 位。
4、日期类型
Date 日期（日-月-年） DD-MM-YY(HH-MI-SS)
Timestamp 跟 date 比 它可以精确到微秒。精确范围 0~9 默认为 6.

八、id、rowid、rownum 的区别

rowid 物理位置的唯一标识。

而 id 是逻辑上的唯一标识，所以 rowid 查找速度要快于 id,是目前最快的

定位一条记录的方式

rowid 和 rownum 都是"伪数列"

所谓“伪数列”也就是默认隐藏的一个数列。

rownum 用于标记结果集中结果顺序的一个字段，

它的特点是按顺序标记，而且是连续的，

换句话说就是只有有 rownum=1 的记录，才可能有 rownum=2 的记录。

rownum 关键字只能和<或者<=直接关联

如果是>或者=则需要给他起个别名

九、主键和唯一索引的区别？

在创建主键的同时会生成对应的唯一索引,主键在保证数据唯一性的同时不允许为空，而唯一可以有一个为空数据项，一个表中只能有一个主键，但是一个主键可以有多个字段，一个表中可以有多个唯一索引。

十、Preparedstatement 和 statement 的区别

用 Prepared statement 进行开发。Prepared statement 是预编译的，而 statement 不是，在每次执行 sql 语句的增删改时，如果是一条数据两者没差距，但如果数据量大于 1，那么每次执行 sql 语句 statement 都要重新编译一次，而 Prepared statement 不用，Prepared statement 的运行效率大于 statement；从代码的可维护性和可读性来说，虽然用 Prepared statement 来代替 statement 会使代码多出几行，但这样的代码无论从可读性还是可维护性来说，都比直接使用 statement 的代码高很多档次；最重要的一点，从安全角度来说，使用 Prepared statement 可以大大提高程序的安全性，因为 Prepared statement 是用‘?’传参,可以防止 sql 注入，具有安全性，而 statement 用的是‘+’字符串拼接，安全性较低。

十一、数据库三范式

第一范式：数据库表中的所有字段值都是不可分解的原子值。
第二范式：需要确保数据库表中的每一列都和主键相关，而不能只与主键的某一部分相关（主要针对联合主键而言）
第三范式：需要确保数据表中的每一列数据都和主键直接相关，而不能间接相关

十二、视图概述

视图可以视为“虚拟表”或“存储的查询”
创建视图所依据的表称为“基表”
视图的优点：
提供了另外一种级别的表安全性:隐藏了一些关键的字段
简化的用户的 SQL 命令
隔离基表结构的改变

十三、存储过程概述

存储过程（Stored Procedure）
可以包含逻辑判断的 sql 语句集合。
是经过预编译，存在于数据库中。
通过调用指定存储过程的名字（可有参，可无参）来执行。
优点：
简化了复杂的业务逻辑，根据需要可重复使用
屏蔽了底层细节，不暴露表信息即可完成操作

降低网络的通信量，多条语句可以封装成一个存储过程来执行
设置访问权限来提高安全性
提高执行效率，因为它是预编译以及存储在数据库中

缺点：

可移植性差，相同的存储过程并不能跨多个数据库进行操作
大量使用存储过程后，首先会使服务器压力增大，而且维护难度逐渐增加

存储过程的语法：

```
--下面是在 oracle 数据库下最基本的语法
--仅创建一个名为 testProcedure  的无参的存储过程
--IS 也可以是 AS
--如果已经存在名为  testProcedure  的存储过程，下面的语法会出现  名称已被使用的错误
--解决办法：
--第一句可以写成  create or replace procedure testProcedure
--这样会替换原有的存储过程
--NULL 表示任何可以正确执行的 sql  语句，但至少一句

create procedure testProcedure
IS
BEGIN

NULL

END;
```

存储过程的参数的分类:

IN
OUT
INOUT

注意：

存储过程之间可相互调用
存储过程一般修改后，立即生效。

十四、 索引概述

1、索引的概念

索引就是加快查询表中数据的方法。

数据库的索引类似于书籍的索引。

在书籍中，索引允许用户不必翻阅完整个书就能迅速地找到所需要的信息。

在数据库中，索引也允许数据库程序迅速地找到表中的数据，

而不必扫描整个数据库.

2、索引的优点

1.创建唯一性索引，保证数据库表中每一行数据的唯一性

2.大大加快数据的检索速度，这也是创建索引的最主要的原因

3.减少磁盘 IO（向字典一样可以直接定位）

3、索引的缺点

1.创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加

2.索引需要占用额外的物理空间

3.当对表中的数据进行增加、删除和修改的时候，

索引也要动态的维护，降低了数据的维护速度

4、索引的分类

1.普通索引和唯一性索引

普通索引：CREATE INDEX mycolumn_index ON mytable (myclumn)

唯一性索引：保证在索引列中的全部数据是唯一的

CREATE unique INDEX mycolumn_index ON mytable (myclumn)

2. 单个索引和复合索引

单个索引：对单个字段建立索引

复合索引：又叫组合索引，在索引建立语句中同时包含多个字段名，最多 16 个字段

CREATE INDEX name_index ON userInfo(firstname,lastname)

3.顺序索引，散列索引,位图索引

十五、必背的 sql 语句

一：oracle 分页

select * from (select t.*, rownum rn from (select * from menu order by id desc) t where rownum < 10) where rn >=5

二: mysql 分页

select * from music where id limit 5,5

三：oracle 中如何快速将一张表的数据复制到另外一张表中(另外一张表不存在，另外一张　　表存在，但数据为空)

1、.不存在另一张表时：

create　table　新表　as　select * from 将要复制的表

2、存在另一张表时：

insert　into　新表名　select　字段　from　将要复制的表名

四：音乐专辑

查询出 special <app:ds:special>表中的 id　专辑名 并下面有多少首歌曲

Select　s.id , min(s.sname),count(m.mid)　from　special s　inner join　ms　m　on　s.id=m.id　group　by　s.id

五：快速删除一张表（不可事物回滚，也就是没有日志记录）

TRUNCATE from 表名

六：inner join

select 查找信息 from 表名 1 inner join 表名 2 on 表名 1.列名 = 表名 2.列名

七：left join

左外连接　　select 查找信息 from 表名 1 left join 表名 2 on 表名 1.列名 = 表名 2.列名

八：right join

右外连接　select 查找信息 from 表名 1 right join 表名 2 on 表名 1.列名 = 表名 2.列名

九：oracle 中查询遍历树形结构（start with）

```
select * from extmenu
start with pid=0
connect by prior id = pid
```

十：查询出来 60-70,80-90,95-100 学生的信息

select * from stu where chengji between 60 and 70 or between 80 and 90 or between 95 and 100

select * from stu where chengji > 60 and chengji < 70 or chengji > 80 and chengji < 90 or chengji > 95 and chengji < 100

十一：用 exists 替换 in------进行联表查询

select * from dept where exists(select * from emp where emp.deptno=dept.deptno);

或

select * from dept d inner join emp e on d.deptno = e.deptno(只查询出两表共同拥有的字段数据)

十二：删除表中的重复数据：

```
delete from xin a where a.rowid != (
        select max(b.rowid) from xin b
            where a.name = b.name
    );
```

业务场景篇

一、Spring 的概述

Spring 是完全面向接口的设计，降低程序耦合性，主要是事务控制并创建 bean 实例对象。在 ssh 整合时，充当黏合剂的作用。IOC(Inversion of Control) 控制反转/依赖注入，又称 DI(Dependency Injection) (依赖注入)

IOC 的作用：产生对象实例，所以它是基于工厂设计模式的

Spring IOC 的注入

通过属性进行注入，通过构造函数进行注入，

注入对象数组　　注入 List 集合

注入 Map 集合　　注入 Properties 类型

Spring IOC 自动绑定模式：

可以设置 autowire 按以下方式进行绑定

按 byType 只要类型一致会自动寻找，

按 byName 自动按属性名称进行自动查找匹配.

AOP 面向方面（切面）编程
AOP 是 OOP 的延续，是 Aspect Oriented Programming 的缩写，
意思是面向方面(切面)编程。
注：OOP(Object-Oriented Programming） 面向对象编程

AOP 主要应用于日志记录，性能统计，安全控制,事务处理（项目中使用的）等方面。

Spring 中实现 AOP 技术：
在 Spring 中可以通过代理模式来实现 AOP
代理模式分为
静态代理：一个接口，分别有一个真实实现和一个代理实现。
动态代理：通过代理类的代理，接口和实现类之间可以不直接发生联系，而可以在运行期（Runtime）实现动态关联。

动态代理有两种实现方式，可以通过 jdk 的动态代理实现也可以通过 cglib 来实现而 AOP 默认是通过 jdk 的动态代理来实现的。jdk 的动态代理必须要有接口的支持，而 cglib 不需要，它是基于类的。

Spring AOP 事务的描述：
在 spring-common.xml 里通过<aop:config>里面先设定一个表达式 ,设定对 service 里那些方法 如：对 add* ,delete*,update*等开头的方法进行事务拦截。我们需要配置事务的传播（propagation="REQUIRED"）特性,通常把增,删,改以外的操作需要配置成只读事务（read-only="true"）.只读事务可以提高性能。之后引入 tx:advice,在 tx:advice 引用 transactionManager（事务管理）,在事务管理里再引入 sessionFactory,sessionFactory 注入 dataSource，最后通过<aop:config> 引入 txAdvice。

Spring 实现 ioc 控制反转描述：

原来需要我们自己进行 bean 的创建以及注入，而现在交给 spring 容器去完成 bean 的创建以及注入。

所谓的“控制反转”就是 对象控制权的转移，
从程序代码本身转移到了外部容器。

官方解释:
控制反转即 IoC (Inversion of Control)，
它把传统上由程序代码直接操控的对象的调用权交给容器，
通过容器来实现对象组件的装配和管理。
所谓的“控制反转”概念就是对组件对象控制权的转移，
从程序代码本身转移到了外部容器。

二、事务概述

在数据库中,所谓事务是指一组逻辑操作单元即一组 sql 语句。当这个单元中的一部分操作失败,整个事务回滚，只有全部正确才完成提交。

事务的 ACID 属性

1. 原子性（Atomicity）
   原子性是指事务是一个不可分割的工作单位，事务中的操作要么都发生，
   要么都不发生。

2. 一致性（Consistency）
   事务必须使数据库从一个一致性状态变换到另外一个一致性状态。(数据不被破坏)

3. 隔离性（Isolation）
   事务的隔离性是指一个事务的执行不能被其他事务干扰.

4. 持久性（Durability）

   持久性是指一个事务一旦被提交，
   它对数据库中数据的改变就是永久性的.

   在 JDBC 中，
   事务默认是自动提交的，
   每次执行一个 SQL 语句时，如果执行成功，
   就会向数据库自动提交，而不能回滚

为了让多个 SQL 语句作为一个事务执行：

（1）执行语句前调用 Connection 对象的 setAutoCommit(false);
以取消自动提交事务

（2）在所有的 SQL 语句都成功执行后，调用 commit(); 方法提交事务

（3）在出现异常时，调用 rollback(); 方法回滚事务。

三、权限概述

权限涉及到 5 张表：
用户表，角色表，权限表(菜单表)，用户角色关联表，角色权限关联表

当用户登录时，根据用户名和密码到用户表验证信息是否合法，如果合法
则获取用户信息，之后根据用户 id 再到用户角色关联表中得到相关连的角色
id 集合,之后根据角色 id 再到角色权限关联表中获取该角色所拥有的权限 id 集合，
然后再根据权限 id 集合到权限表（菜单表）中获取具体的菜单，展现给当前
登录用户，从而达到不同用用户看到不同的菜单权限。

我们通过 ZTree 来给角色赋权并且通过 ZTree 来展示菜单，以及通过 ZTree 来管理菜单即增加和编辑菜单。

我们做的权限控制到 url 级别,为了防止用户不登录直接输入 url 访问的这个弊端，通过拦截器进行拦截验证。

四、OSCache 业务场景

在我以前的项目中，我们考虑了系统性能问题，这个时候我们采用了 Oscache 缓存，刚开始把这个功能交给了项目组中的另外一个同事来做的，但是他做完的时候他发现缓存中明明已经缓存了数据，但是在取得时候发现没有数据，我们项目经理让我去帮忙看看这个问题，我阅读完他的代码之后，我发现了他每次缓存的时候都是调用一个新的缓存对象的方法，结果出现了明明已经走了缓存的方法而取不到数据的问题，通过我多年的工作经验，我就想到了应该用单例模式去封装一个单例工具类来调用 oscache。但是，在后来的测试过程中，发现当并发访问的时候也会出现上述的问题，这个时候我直接采取的 DCL（双重判定锁）单例模式封装了工具类，既解决了线程安全问题，相对的性能问题也考虑到了，这个问题才得到了完善的解决。

五、线程概述

线程的状态以及状态之间的相互转换：

1、新建状态(New)：新创建了一个线程对象。

2、就绪状态(Runnable)：线程对象创建后，其他线程调用了该对象的 start()方法。该状态的线程位于可运行线程池中，变得可运行，等待获取 CPU 的使用权。

3、运行状态(Running)：就绪状态的线程获取了 CPU，执行程序代码。

4、阻塞状态(Blocked)：阻塞状态是线程因为某种原因放弃 CPU 使用权，暂时停止运行。直到线程进入就绪状态，才有机会转到运行状态。阻塞的情况分三种：

(一)、等待阻塞：运行的线程执行 wait()方法，JVM 会把该线程放入等待池中。

(二)、同步阻塞：运行的线程在获取对象的同步锁时，若该同步锁被别的线程占用，则 JVM 会把该线程放入锁池中。

(三)、其他阻塞：运行的线程执行 sleep()或 join()方法，或者发出了 I/O 请求时，JVM 会把该线程置为阻塞状态。当 sleep()状态超时、join()等待线程终止或者超时、或者 I/O 处理完毕时，线程重新转入就绪状态。

5、死亡状态(Dead)：线程执行完了或者因异常退出了 run()方法，该线程结束生命周期。

实现线程的两种方式：

是继承 Thread 类或实现 Runnable 接口，但不管怎样，当 new 了这个对象后，线程就已经进入了初始状态

wait 和 sleep 的区别：

线程访问：

锁池状态，之后等待锁释放，然后访问代码

wait

等待队列(释放资源)--->调用 notify 或者 notifyall 之后锁池状态--->(等待锁释放)--->可运行状态--->运行状态---->访问代码

sleep,join

不释放资源-->结束后直接进入可运行状态--->运行状态---->访问代码

一个 java 控制台程序，默认运行两个线程，一个主线程，一个垃圾回收线程。

线程与进程的区别：

1.线程(Thread)与进程(Process)

进程定义的是应用程序与应用程序之间的边界，通常来说一个进程就代表一个与之对应的应用程序。不同的进程之间不能共享代码和数据空间，而同一进程的不同线程可以共享代码和数据空间。

2.一个进程可以包括若干个线程，同时创建多个线程来完成某项任务，便是多线程。

六、Ajax 请求 Session 超时问题

我在做项目时有时会遇到 session 超时问题，如果 session 超时，平常请求没有什么问题，通过拦截器可以正确跳到登陆页面，可是你如果用 ajax 请求的话这就出现问题了，因为 ajax 是异步的，局部刷新，所以登陆界面不会再全页面中显示，他只会显示到页面的一部分当中。所以根据我这几年的经验找到了我认为比较好的一种方法。因为那我用的框架是和 struts2 集成的，所以就在拦截器中进行设置：

首先判断 session 是否为空就是判断 session 是否超时，如果超时就取出请求的 head 头信息 request.getHeader("x-requested-with")，如果不为空就和 XMLHttpRequest(Ajax 标识)进行比较(request.getHeader("x-requested-with").equalsIgnoreCase("XMLHttpRequest"))) 如果相等说明此请求是 ajax 请求。

如果是 ajax 请求就可以用 response.setHeader("键","值")来设置一个标识来告诉用户这是 ajax 请求并且 session 超时时发出的，这样我就可以在回调函数中取出自己设置的那个唯一标识：XMLHttpRequest.getResponseHeader("");如果取出的值是和自己在后台中设置的值一样的话，就证明 session 已经超时，这样就可以设置 window.location.replace("登陆界面")，来跳转到登陆界面了。

这样做虽然解决了问题，但是，会在每个回调函数中写入那些代码，这样的话代码就会显得特别零散，所以就想能不能定义一个全局的设置所以就找到了 jqery 的 ajaxSetUp 方法，通过 ajaxSetUp 对 jqery 的 ajax 进行全局的判断(ajaxSetUp 就相当于 ajax 的拦截器)，通过设置 ajaxSetUp 里的 complete，它就相当于回调函数，这样那就弥补了上一方法的不足。我做项目时还用到 $(document).ajaxStart()，这是 ajax 请求时的事件；$(document).ajaxSuccess()，这是 AJAX 请求成功后的事件。我一般用他们来显示遮罩层和隐藏遮罩层用的加遮罩层是为了不让用户重复提交，更提高了用户体验度，让用户知道已经提交了。

七：java 线程池概述

ava 线程池的工作原理和数据库连接池的差不多，因为每次重新创建线程都是很耗资源的操作，所以我们可以建立一个线程池，这样当需要用到线程进行某些操作时，就可以直接去线程池里面找到空闲的线程，这样就可以直接使用，而不用等到用到的时候再去创建，用完之后可以把该线程重新放入线程池供其他请求使用从而提高应用程序的性能。

八、OSCache 概述

oscache 是一个高性能的 j2ee 框架，可以和任何 java 代码进行集成，并且还可以通过标签对页面内容进行缓存，还以缓存请求。
我们通常将那些频繁访问但是又不是经常改变的数据进行缓存。为了保证缓存数据的有效性，在数据发生改变的时候，我们要刷新缓存，避免脏数据的出现。刷新缓存的策略有两种，一种是定时刷新，一种手动刷新。

缓存数据的时机通常也分为两种 ,即在 tomcat(web 容器)启动时候加载数据进行缓存，另外也可以在用户第一次访问数据的时候进行缓存，这个相当于缓存的立即加载和按需加载。

缓存的层次如下:jsp-->action-->service-->dao,缓存越靠前对性能的提升越大

一个 action 里面可以有多个 service,一个 service 中可以有多个 dao 或者多个 service

任何类之间都可以进行相互调用，可以通过构造函数传参，set,get 传参或者是方法传参将相关的类连接起来。

九、OSCache+autocomplete+单例业务场景

在我以前做某项目的过程中 ,其中我们在做产品列表的查询的时候为了提高用户的体验度，我们使用了 autocomplete 插件来代替 select 进行品牌的选择，才开始的时候每次都要根据用户输入的信息去查询数据库进行模糊匹配返回结果，后来我们考虑到系统的性能，因此我们采用了 oscache 缓存，才开始这个功能是交给我们项目组中的另外一个同事来做的，但是他做完后，我们在使用这个工具类的时候，发现有时缓存中明明已经有时我们需要的数据，但是从缓存里面取的时候，发现没有，之后项目经理让我去帮这个同事看看这个问题，我经过阅读他的代码发现，它里面在使用缓存的时候，针对于每次方法的调用都产生一个新的实例，结果导致了上面的问题，这个时候我想起了可以使用设计模式中的单例模式来解决这个问题，才开始我直接采用了普通的单列模式，但是后来在测试的过程中，发现当用户并发量大的时候还是会出现上面的问题，之后我再次考虑了代码，最后发现是因为没有给单列模式加锁的原因，从而导致了大用户并发的时候，线程安全的问题，之后我便在方法上加上了 synchronized 关键字，解决上述的问题，但是后来测试人员反馈，觉的这段的

性能有问题，我考虑之后便采用在方法体内加锁并结合双重判定的方式解决了上面的问题。我们是将数据在 tomcat 启动的时候加载到缓存中，之后用户进行查询的时候直接从缓存中获取数据，根据前缀匹配进行查询，将结果返回给用户。这样在提高用户体验度的同时也提高性能。

十、缓存概述

应用程序为了提高性能，可以通过使用缓存来达到目的，缓存的存储介质可以内存或者硬盘，通常将数据存储在内存里，确切的说是 jvm 的内存中，缓存是基于 Map 这种思想构建的，以键值对的方式进行存储和获取，之所以还可以将缓存的数据存储在硬盘中，是因为内存资源相当有限和宝贵，所以当内存资源不足的时候，就可以将其存储到硬盘中，虽然硬盘的存取速度比内存要慢，但是因为减少了网络通信量，所以还是提高程序的性能。缓存可以分为客户端缓存和服务器端缓存，所谓的客户端缓存通常指的是 IE 浏览器的缓存，服务器端缓存指的 web 服务器的缓存，通常可以通过第三方组件实现，如 oscache,memcache。

十一、实现页面静态化业务场景

我们在做某项目时，涉及到程序访问的性能问题，这时候我们想到可以通过静态化来提高用户访问时候的性能，所以我们就采用了 freemarker 模板引擎，考虑到页面也是要有动态的变化的，所以我们采用 spring 定时器在每天晚上 2 点钟的时候定时再次生成 html 静态页面，考虑发布时候的性能问题，我们又采取线程池技术，让多个线程同时发布，从而缩减发布时间。

十二、servlet 线程安全描述

servlet 是单列的，对于所有请求都使用一个实例，所以如果有全局变量被多线程使用的时候，就会出现线程安全问题。

解决这个问题有三种方案:

1.实现 singleThreadModel 接口，这样对于每次请求都会创建一个新的 servlet 实例，这样就会消耗服务端内存，降低性能，但是这个接口已经过时，不推荐使用。

2.可以通过加锁(synchroniezd 关键字)来避免线程安全问题。这个时候虽然还是单列，但是对于多线程的访问，每次只能有一个请求进行方法体内执行，只有执行完毕后，其他线程才允许访问，降低吞吐量。

3.避免使用全局变量，使用局部变量可以避免线程安全问题，强烈推荐使用此方法来解决 servlet 线程安全的问题。

十三、(jbpm4)工作流引擎描述:

JPBM 是 JBOSS 旗下的一个开源的基于 hibernate 的工作流引擎。工作流就是在日常生活中，我们一些常见的如请假流程、采购流程、入职流程，通俗的来讲就是一些在现实生活中的流程以信息化以程序的方式实现。

一个工作流首先需要进行流程定义，流程定义是由节点和跳转组成的，节点又可以称为环节、活动节点、活动环节，并且节点也可以分为两大类型：人工节点和自动节点，人工节点有 start 开始节点、end 结束节点、task 任务节点，自动节点有 decision 判断节点、fork 分支节点、join 聚合节点和 state 状态节点，并且一个流程有且只有一个开始节点，但可以有多个结束节点。

流程定义是静止的，它在运行状态时会转换成流程实例，一个流程定义可以对应多个流程实例。流程运行后，会产生两个文件，*.jdpl.xml 文件和*.png 图片文件，也会生成 18 张数据库表，常用且核心的表有 JBPM4_LOB 存储表，主要存储 xml 文件和 png 图片、JBPM4_TASK 任务表、JBPM4_EXECUTION 流程实例表、JBPM4_VARIABLE 变量表。

JBPM 有五大核心类：

ProcessEngine：主要获取各种的 Service
RepositoryService：主要发布流程定义
ExecutionService：主要操作流程实例
TaskService：主要操作人工服务
HistoryService：主要操作历史服务。

核心方法：

读取 jbpm 定义的文件生成 zip 包存到 lob 表中：createDeployment()
获取流程定义列表：createProcessDefinitionQuery
根据定义的 key 或 id 来启动流程实例：startProcessInstanceByKey(id)
获取待办任务列表：findPersonalTasks(userName)
完成指定任务列表：completeTask(*.getActivityId())
获取历史任务列表：createHistoryTaskQuery()
获取流程实例的 ID：task.getExecutionId()

(了解的表)
JBPM4_HIST_ACTINST 流程活动(节点) 实例表
JBPM4_HIST_DETAIL 流程历史详细表
JBPM4_HIST_PROCINST 流程实例历史表
JBPM4_HIST_TASK 流程任务实例历史表
JBPM4_HIST_VAR 流程变量(上下文) 历史表

十四、JPBM 业务场景

首先进行请假的流程定义，我们流程的定义是（员工提交请假单---》经理审批---》总监审批---》总经理审批---》结束），通过 repositoryService 将其发布部署到 jbpm4_lob 表中，

之后获取流程定义列表，选中请假的流程定义，员工开始进行请假单的填写，保存并通过 executionService 开启流程实例，然后用 taskService 获取经理的待办任务列表，选中待办任务，进行审批，通过调用 taskService.completeTask()进入到总监审批环节，然后用总监进行登录，同样获取待办任务列表，然后调用 taskService.completeTask()进入总经理审批环节，总经理审批之后，结束流程。在这个过程中我们还可以根据 historyService 查看当前登录人已办的任务列表。

十五、Ant 描述

Ant 是 apache 旗下的对项目进行自动打包、编译、部署的构建工具，他主要具有　　轻量级并且跨平台的特性，而且基于 jvm，默认文件名为 build.xml

Ant 主要的标签：

Project 根标签，

target 任务标签，

property 属性标签，自定义键/值 供多次使用，

java 执行编译后的 java 文件

javac 编译 java 文件

war 打成 war 包

其它标签：copy，delete，mkdir，move，echo 等。

十六、FreeMarker 描述

FreeMarker 是一个用 Java 语言编写的模板引擎，它是基于模板来生成文本输出的通用工具。Freemarker 可以生成 HTML， XML，JSP 或 Java 等多种文本输出。

工作原理：定义模板文件，嵌入数据源，通过模板显示准备的数据

（数据 + 模板 = 输出）

我们在使用模板中发现 freemarker 具有许多优点，它彻底的分离表现层和业务逻辑，模板只负责数据在页面中的表现，不涉及任何的逻辑代码，所以使得开发过程中的人员分工更加明确，作为界面开发人员，只需专心创建 HTML 文件、图像以及 Web 页面的其他可视化方面，不用理会数据；而程序开发人员则专注于系统实现，负责为页面准备要显示的数据。如果使用 jsp 来展示，开发阶段进行功能调适时，需要频繁的修改 JSP，每次修改都要编译和转换，浪费了大量时间，FreeMarker 模板技术不存在编译和转换的问题，在开发过程中，我们在不必在等待界面设计开发人员完成页面原型后再来开发程序。由此使用 freemarker 还可以大大提高开发效率。

十七、webService 描述

webservice 是 SOA（面向服务编程）的一种实现，主要是用来实现异构平台通信也就 是不同平台不同项目之间的数据传输，从而避免了信息孤岛的问题，它之所以能够进行异构平台通信是因为它是完全基于 xml 的，所以说，webService 是跨平台，

跨语言，跨框架的。

webservice 的三要素分别是：

wsdl（webservice description language）用来描述发布的接口（服务）

soap(simple object access protocol) 是 xml 和 http 的结合，是 webservice 数据通信

的协议

uddi（univers）用来管理,分发,查询 webService 的服务

webservice 的具体三种实现方式（框架）

1. Axis2:可以用多种语言开发（java c c++ C#），是一个重量级框架，功能非常 强 大，所以说涉及到的 api 非常多，所以说它的性能比较慢。

2. Xfire：它相比 Axis2 来说是一个轻量级框架，但是它的性能要比 Axis2 高，Api 也相对少。

3. cxf：是 Xfire 的升级版，就好比是，struts2 是 webwork 的升级，人们会逐渐淡忘 webwork,但是 struts2 会慢慢变成主流。然而 cxf 就好比是 struts2，会成为主流技术。然后 cxf 和 spring 集成起来非常方便，简易，性能方面也要比 Xfire 高。

【注】jdk6 自带的 webservice jws

业务场景

我在以前做项目的时候，其中遇到一个功能，需要进行两个项目之间的数据的传输，项目经理让我去完成这个任务，我根据以往的项目经验，想到两种解决方案，第一种就是开放另外一个项目的数据库的权限给我，然后我直接通过访问另外一个项目的数据库，来得到需要的信息，但后来我分析了下，觉的这种方式不安全，而且因为当时这个项目是另外一家公司负责在做，所以数据库里面的表结构，以及以后牵涉到的责任问题都很多，所以我就采用了第二种方案，即通过 webservices 的方式，进行异构系统之间数据信息的传递，webservices 的具体实现，有 xfire,cxf,axis2,我根据以往的项目经验，了解到 cxf 是 xfire 的升级版本，适用于 java 语言，xfire/cxf 性能比 axis2 要高，并且和 spring 整合起来也比较方便，而 axis2 支持更多的 语言，性能相对于 cxf 要低，通过上面分析，结合我们目前的两个项目都是基于 java 语言的，所以我采用 cxf 这种方式实现了两个项目之间数据的传递，我们为了保证 webservice 的安全性我们采用了基于 WS-Security 标准的安全验证(使用 CXF 回调函数)。

webservice 服务端配置流程

首先在 web.xml 中引入 cxfServlet 核心类，指定对以/cxf 开头的 url 路径提供 webservice 服务，之后我们在要发布成 webservice 接口上添加@Webservice 注解，而且还要在实现类上添加同样的 webservice 注解并且要说明实现了哪个接口，之后在 spring-webservice.xml 中发布 webservice 服务，通过 jaxws:endpoint 这个标签，并且

在标签配置 implementor 和 address 来表明实现服务的类，以及发布的地址，最后在浏览器中输入相关的 webservice 地址?wsdl 来验证服务是否发布成功。

webservice 客户端的配置

首先通过 wsdl2java 根据发布的 webservice 服务端地址的 wsdl 生成客户端调用的中间桥梁 java 类，将生成的 java 类拷贝到客户端项目中，配置 spring-client.xml 文件，通过 jaxws:client 定义一个 bean,并通过 address 属性指明要访问的 webservice 的服务地址，通过 serviceClass 指明充当中间桥梁的服务类，之后获取该 bean,就可以通过它来访问发布的 webservice 接口中的方法。

十八、oracle 索引概述

索引呢　是与表相关的一个可选结构，可以提高 sql 语句的检索效率，相当于我们的字典目录　，可以快速进行定位　，所以可以减少磁盘 I/O,　但是因为索引在物理与逻辑上都是独立于表的数据　它会占用一定的物理空间(额外磁盘空间)　所以并不是索引越多越好,而我们应该根据业务需求去创建索引,而且进行增删改操作时　oracle　又要自动维护索引所以在一定程度上也降低了维护速度，而且我们在创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加 ,我们一般创建索引呢　是这样创建的　create　index　索引名　on　表名(字段)，索引又分为普通索引　唯一索引(unique)　单个索引　复合索引(又叫组合索引，在索引建立语句中同时可包含多个字段名)，顺序索引，散列索引,位图索引。

十九、oracle 存储过程

存储过程就是封装一些 sql 的集合，也就是一条条的 sql 语句，过程的优点就是简化了 sql 命令加上它是预编译的，所以它的执行效率和性能较高，再者，如果不调用过程的话就要和数据库发生多次交互，调用过程只需传一个命令所有的那些执行逻辑都在数据库端执行，所以说它降低了网络的通信量，其次，存储过程大大提高了安全性，这就是优点

缺点呢，就是不同的数据库对过程支持的关键字支持的关键字都是不一样的，所以它的移植性是非常差的，再者，它的维护性难度也比较大，因为它没有专业的调试和维护工具，所以说它维护起来比较麻烦，这就是存储过程的基本概述.

二十、Junit 业务场景

在我们开发项目的时候为了提高代码的性能和保证逻辑正确性，在我们编写代码后往往都要进行单元测试，来验证代码，当时我们公司开发人员全部使用的 main 方法来进行验证，但是使用 mian 的最大缺点就是不能将多个类同时进行验证，验证的结果不直观，测试复杂（每个类都要写 main 方法，单个运行），一定程度上浪费时间，所有我和项目经理提议使用专业测试工具 Junit 来进行测试，因为 Junit 是一个 Java 语言的单元测试框架 ，测试简单，不仅可以提供工作效率和代码的质量，也提高团队的合作能力，我提议后我们进行了 Junit 的培训使用 Junit4 加注解的方式来测试。

二十一、Apache+Tomcat 实现负载均衡及 seesion 复制

当我们 tomcat 访问量大,线程连接数不够时,我们考虑到了 tomcat 的负载均衡来分担过多的访问.性能方面负载均衡也能利用多台 tomcat 来增大内存量,

流程,准备工作 apache,Jk_mod,tomcat,在 apache 的 conf/httpd.conf 文件中 使用 include 标签引入我们自定义的一个 mood_jl.conf,在 modules 中引入下载的 k_mod-apache-X.X.XX.so 文件,在其中引入我们的.so,及 work.properties 文件,及指定负载分配控制器 controller, 在 work.properties 文件中 worker.list=controller,tomcat1,tomcat2 指定 service,worker.tomcat1.port Ajp 端口号,type 是 ajp,host 为指定 ip,lbfactor 指定分配权重值越大分担请求越多 ,worker.controller.type=lbworker.controller.balanced_workers=tomcat1,tomcat2 指定分担请求的 tomcat Session 的复制在 tomcat 中 service.xml 中 Engine 标签加入 jvmRoute 值为 work,properties 中指定的 tomcat 名称,然后打开<Cluster 标签的注释,最后在应用中程序的 web.xml 文件中增加<distributable/>。

二十二、Ant 业务场景

Ant 是基于 java 语言编写的，因此具有跨平台的特性，此外还具有简洁方便，灵活配置的特性，因此我就在 XX 项目中使用 ant 进行项目的编译，打包，部署操作。使用 ant 之后，如果我们在客户那里修改代码后，就可以直接使用 ant 进行编译，打包，部署，而不需要为了编译，打包，部署专门在客户那里安装 eclipse.此外使用 ant 也可以直接和 svn 进行交互，下载源码的同时进行编译，打包，部署。

二十三、maven 业务场景

前段时间在研究 maven，知道 maven 是一个项目管理工具，其核心特点就是通过 maven 可以进行包的依赖管理，保证 jar 包版本的一致性，以及可以使多个项目共享 jar 包，从而能够在开发大型 j2ee 应用的时候，减小项目的大小，并且和 ant

比起来，maven 根据“约定优于配置”的特性，可以对其项目的编译打包部署进行了更为抽象的封装，使得自己不需要像 ant 那样进行详细配置文件的编写，直接使用系统预定好的 mvn clean,compile,test,package 等命令进行项目的操作。于是我就在 XX 项目中采用了 maven,为了保证团队中的成员能够节省下载 jar 包所需要的时间，于是我就采用 nexus 搭建了在局域网内的 maven 私服，然后通过配置 settings.xml 中建立 mirror 镜像 ,将所有下载 jar 包的请求都转发到 maven 私服上 ,之后通过在 pom.xml 即(project object model)中配置项目所依赖的 jar 包，从而达到在构建项目的时候，先从本地仓库中查找，如果不存在从内部私服查找，如果不存在最后再从外网 central 服务器查找的机制，达到了节省下载带宽，提高开发效率，以及 jar 包重用的目的。

二十四、Servlet 的概述：

Servlet 是一个 web 容器，我们通常用的 servlet 是 httpservlet，而 httpservlet 又是继承于 genericservlet，而 genericservlet 又实现了 servlet 接口

servlet 的生命周期是 ：先进行实例化，然后是初始化，然后是提高服务，然后销毁，最后不可用，在这五个生命周期，其中，初始化是调用的 init 方法，这个方法只有一个，而提高服务的时候调用的是 service 方法，而我们具体在我们所写的这个方法中，因为我们继承了 httpservlet，其实就是对应了 doGet（ ），doPost(),这种方法，然后据我了解，servlet 是单例的。非线程安全的，我们通常有一下几种方案来解决：

第一种，继承 SingleThreadModel 但是这样每次都会创建一个新的 servlet 实例，但这样消耗服务器的内存，降低了性能，并且这个接口现在已经过时了，不推荐使用。

第二种：我们尽量避免使用全局变量，就我个人而言，我比较喜欢使用这种方法。

第三种，我们可以通过使用 ThreadLocal， 内部结构是一个 Map 结构，用当前线程作为 key,他会创建多个副本。get,set 方法

第四种，我们当然还可以来加锁，进行解决线程问题。

而且我还知道，向我们这种常用的 MVC 框架，struts1，spring 这些 MVC 框架，都是基于 servlet 发展而来的，就比如 struts1 的核心总控制器是 ActionServlet，而 springMVC 的前端总控制器是 dispatchServlet，在项目我们曾经用 serlet 来生成 图片验证码的，防止用户进行暴力破解

（别人问了，再回答）

servlet 的配置文件　web.xml

```
<servlet>
    <servlet-name>ImageCodeServlet</servlet-name>
    <servlet-class>org.leopard.code.ImageCodeServlet</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>ImageCodeServlet</servlet-name>
        <url-pattern>/d</url-pattern>
</servlet-mapping>
```

描述：

我在 web.xml 中，我首先需要写一个 servlet 标签，servlet 标签中有两个子标签，一个叫 servlet-name ,这个 name 可以随便起 ,但是要保证唯一性 ,除此之外 ,在这个 servlet-name 下有一个 servlet-class，这个 servlet-class 对应的就是我后台提高服务的 servlet，除此之

外还有一个 servlet-mapping，这个里边首先有一个 servl-name。，这个 servl-name 首先要保证和上边的 servlet-name 保持一致，除此之外还有一个 url-pattern，这是一个虚拟路径，是用来发送请求的 url 地址

优化篇

一、代码优化

代码结构层次的优化(目的:更加方便代码的维护--可维护性，可读性)

1.代码注释(代码规范)
2.工具类的封装(方便代码的维护，使代码结构更加清晰不臃肿，保证团队里代码质量一致性)
3.公共部分的提取

代码性能的优化(目的:使程序的性能最优化)
1.使用一些性能比较高的类(bufferInputStream)
2.缓冲区块的大小(4k 或者 8k)
3.公共部分的提取
4.通常要用 stringbuffer 替代 string 加号拼接

二、业务优化

我们做项目的时候业务优化这方面最主要是从用户体验度角度进行考虑,减少用户操作的步骤提高工作效率，通常有以下几种：

1.可以通过 tabindex 属性来改变 tab 键盘的操作顺序
2.可以通过回车键来进行搜索或者提交操作
3.对于单选按钮和复选按钮可以通过操作后面的文本来选择前面的单选按钮以及复选按钮
4.添加的信息要按照 id 倒序进行排列
5.进行搜索操作时加入 js loading 操作（不仅告诉用户所进行的请求正在被处理，而且防止用户多次点击提交操作）
6.当进行删除操作的时候要弹出提示框，警告用户要进行删除操作，是否确认。
7.根据 returnURL 在用户登录成功后直接跳到想要访问的资源。
8.进行删除操作时通过 confirm 提示用户是否确认删除操作，操作完后提示操作是否成功。

9.减少用户操作的步骤
10.使用 autocomplete 插件快速进行搜索

必背，必做:

1.可以通过回车键来进行搜索或者提交操作
2.添加的信息要按照 id 倒序进行排列
3.进行搜索操作时加入 js loading 操作（不仅告诉用户所进行的请求正在被处理，而且防止用户多次点击提交操作）
4.当进行删除操作的时候要弹出提示框，警告用户要进行删除操作，是否确认,如果删除成功则弹出提示框告诉用户。
5.减少用户操作的步骤

三、sql 优化

1、SELECT 子句中避免使用 *， 尽量应该根据业务需求按字段进行查询

2、尽量多使用 COMMIT 如对大数据量的分段批量提交释放了资源，减轻了服务器压力

3、在写 sql 语句的话，尽量保持每次查询的 sql 语句字段用大写，因为 oracle 总是先解析 sql 语句，把小写的字母转换成大写的再执行

4、用 UNION-ALL 替换 UNION，因为 UNION-ALL 不会过滤重复数据，所执行效率要快于 UNION,并且 UNION 可以自动排序，而 UNION-ALL 不会

5、避免在索引列上使用计算和函数,这样索引就不能使用

四、防 sql 注入

针对防 sql 注入，我们通常是这样做的：

首先在前台页面对用户输入信息进行 js 验证，对一些特殊字符进行屏蔽，比如：or ,单引号，--，= ，还有就是限制用户名输入的长度，我们一般将其限制在 6---13 位。另外，对于用户的敏感信息我们进行 Md5 加密，还有，为了增加用户体验度和用户友好度，为了不使用户看到一些详细的异常信息我们会进行错误信息页面的定制，像 404,500 错误。另一个我层面讲，这样做也是为了保护我们的一些重要信息。此外，我们会给特定的人分配定定的权限，而不是给其分配管理员权限！

JavaWEB

1、http 的长连接和短连接

HTTP 协议有 HTTP/1.0 版本和 HTTP/1.1 版本。HTTP1.1 默认保持长连接（HTTP persistent connection，也翻译为持久连接），数据传输完成了保持 TCP 连接不断开（不发 RST 包、不四次握手），等待在同域名下继续用这个通道传输数据；相反的就是短连接。

在 HTTP/1.0 中，默认使用的是短连接。也就是说，浏览器和服务器每进行一次 HTTP 操作，就建立一次连接，任务结束就中断连接。从 HTTP/1.1 起，默认使用的是长连接，用以保持连接特性。

1、http 常见的状态码有哪些？

200 OK //客户端请求成功

301 Moved Permanently (永久移除)，请求的 URL 已移走。Response 中应该包含一个 Location URL，说明资源现在所处的位置

302 found 重定向

400 Bad Request //客户端请求有语法错误，不能被服务器所理解

401 Unauthorized //请求未经授权，这个状态代码必须和 WWW-Authenticate 报头域一起使用

403 Forbidden //服务器收到请求，但是拒绝提供服务

404 Not Found //请求资源不存在，eg：输入了错误的 URL

500 Internal Server Error //服务器发生不可预期的错误

503 Server Unavailable //服务器当前不能处理客户端的请求，一段时间后可能恢复正常

3、GET 和 POST 的区别？

（1）GET 请求的数据会附在 URL 之后（就是把数据放置在 HTTP 协议头中），以?分割 URL 和传输数据，参数之间以 & 相连，如：login.action?name=zhagnsan&password=123456。POST 把提交的数据则放置在是 HTTP 包的包体中。

（2）GET 方式提交的数据最多只能是 1024 字节，理论上 POST 没有限制，可传较大量的数据。其实这样说是错误的，不准确的："GET 方式提交的数据最多只能是 1024 字节"，因为 GET 是通过 URL 提交数据，那么 GET 可提交的数据量就跟 URL 的长度有直接关系了。而实际上，URL 不存在参数上限的问题，HTTP 协议规范没有对 URL 长度进行限

制。这个限制是特定的浏览器及服务器对它的限制。IE 对 URL 长度的限制是 2083 字节(2K+35)。对于其他浏览器，如 Netscape、FireFox 等，理论上没有长度限制，其限制取决于操作系统的支持。

（3）POST 的安全性要比 GET 的安全性高。注意：这里所说的安全性和上面 GET 提到的“安全”不是同个概念。上面“安全”的含义仅仅是不作数据修改，而这里安全的含义是真正的 Security 的含义，比如：通过 GET 提交数据，用户名和密码将明文出现在 URL 上，因为(1)登录页面有可能被浏览器缓存，(2)其他人查看浏览器的历史纪录，那么别人就可以拿到你的账号和密码了，除此之外，使用 GET 提交数据还可能会造成 Cross-site request forgery 攻击。

Get 是向服务器发索取数据的一种请求，而 Post 是向服务器提交数据的一种请求，在 FORM（表单）中，Method

默认为"GET"，实质上，GET 和 POST 只是发送机制不同，并不是一个取一个发！

Cookie 和 Session 的区别

Cookie 是 web 服务器发送给浏览器的一块信息，浏览器会在本地一个文件中给每个 web 服务器存储 cookie。以后浏览器再给特定的 web 服务器发送请求时，同时会发送所有为该服务器存储的 cookie。

Session 是存储在 web 服务器端的一块信息。session 对象存储特定用户会话所需的属性及配置信息。当用户在应用程序的 Web 页之间跳转时，存储在 Session 对象中的变量将不会丢失，而是在整个用户会话中一直存在下去。

Cookie 和 session 的不同点：

1、无论客户端做怎样的设置，session 都能够正常工作。当客户端禁用 cookie 时将无法使用 cookie。

2、在存储的数据量方面：session 能够存储任意的java 对象，cookie 只能存储 String 类型的对象。

4、在单点登录中，如果 cookie 被禁用了怎么办？

单点登录的原理是后端生成一个 session ID，然后设置到 cookie，后面的所有请求浏览器都会带上 cookie， 然后服务端从 cookie 里获取 session ID，再查询到用户信息。所以，保持登录的关键不是 cookie，而是通过 cookie 保存和传输的 session ID，其本质是能获取用户信息的数据。除了 cookie，还通常使用 HTTP 请求头来传输。但是这个请求头浏览器不会像 cookie 一样自动携带，需要手工处理。

5、什么是 jsp，什么是Servlet？jsp 和 Servlet 有什么区别？

jsp 本质上就是一个 Servlet，它是 Servlet 的一种特殊形式（由 SUN 公司推出），每个 jsp 页面都是一个 servlet 实例。

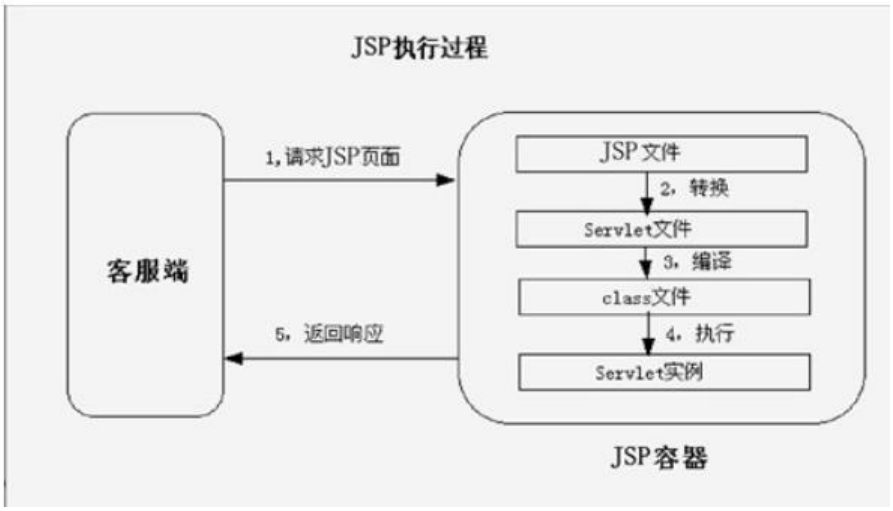
Servlet 是由 Java 提供用于开发 web 服务器应用程序的一个组件，运行在服务端，由 servlet 容器管理，用来生成动态内容。一个 servlet 实例是实现了特殊接口 Servlet 的 Java 类，所有自定义的 servlet 均必须实现 Servlet 接口。

区别：

jsp 是 html 页面中内嵌的 Java 代码，侧重页面显示；

Servlet 是 html 代码和 Java 代码分离，侧重逻辑控制，mvc 设计思想中 jsp 位于视图层，servlet 位于控制层

Jsp 运行机制：如下图



JVM 只能识别 Java 类,并不能识别 jsp 代码 !web 容器收到以.jsp 为扩展名的 url 请求时,会将访问请求交给 tomcat 中 jsp 引擎处理,每个 jsp 页面第一次被访问时,jsp 引擎将 jsp 代码解释为一个 servlet 源程序，接着编译 servlet 源程序生成.class 文件，再有 web 容器 servlet 引擎去装载执行 servlet 程序，实现页面交互。

6、servlet 生命周期

Servlet 加载—>实例化—>服务—>销毁。

（1）生命周期详解：

init（）:

在 Servlet 的生命周期中,仅执行一次 init()方法。它是在服务器装入 Servlet 时执行的，负责初始化 Servlet 对象。可以配置服务器，以在启动服务器或客户机首次访问 Servlet 时装入 Servlet。无论有多少客户机访问 Servlet，都不会重复执行 init（）。

service（）:

它是 Servlet 的核心,负责响应客户的请求。每当一个客户请求一个 HttpServlet 对象，该对象的 Service()方法就要调用，而且传递给这个方法一个“请求”（ServletRequest）对

象和一个“响应”（ServletResponse）对象作为参数。在 HttpServlet 中已存在 Service()方法。默认的服务功能是调用与 HTTP 请求的方法相应的 do 功能。

destroy（ ）:

仅执行一次，在服务器端停止且卸载 Servlet 时执行该方法。当 Servlet 对象退出生命周期时 ,负责释放占用的资源。一个 Servlet 在运行 service()方法时可能会产生其他的线程，因此需要确认在调用 destroy()方法时，这些线程已经终止或完成。

（2）如何与 Tomcat 结合工作步骤：

1、Web Client 向 Servlet 容器（Tomcat）发出 Http 请求

2、Servlet 容器接收 Web Client 的请求

3、Servlet 容器创建一个 HttpRequest 对象，将 Web Client 请求的信息封装到这个对象中。

4、Servlet 容器创建一个 HttpResponse 对象

5、Servlet 容器调用 HttpServlet 对象的 service 方法，把 HttpRequest 对象与 HttpResponse 对象作为参数传给 HttpServlet 对象。

6、HttpServlet 调用 HttpRequest 对象的有关方法，获取 Http 请求信息。

7、HttpServlet 调用 HttpResponse 对象的有关方法，生成响应数据。

7、servlet 特性

单例多线程

8、转发和重定向的区别(forward()和 sendRedirect()的区别)

转发在服务器端完成的；重定向是在客户端完成的

转发的速度快；重定向速度慢

转发的是同一次请求；重定向客户端发送两次不同请求

转发不会执行转发后的代码；重定向会执行重定向之后的代码

转发地址栏没有变化；重定向地址栏有变化

转发必须是在同一台服务器下完成；重定向可以在不同的服务器下完成

转发可以使用 request 域共享数据；重定向是两次请求不可以

9、Cookie 过期和 Session 超时的区别

Cookie 过期：

Cookie 由浏览器管理，如果 Cookie 有过期时间，浏览器会在过期时间将其销毁。如果没有设置过期时间浏览器关闭后 Cookie 会过期

Session 超时：

Session 对象由服务器管理，服务器会计算 Session 对象的不活动时间，如果 Session 对象设置了超时时间，Session 对象超过此时间会被销毁。如果没有设置默认 30 分钟会被销毁(服务器默认配置)。

10、jquery 如何发送 ajax 请求?

常见方法：

1、 $.get(),$.post()

两个方法使用基本一样，根据需要传入四个参数，请求地址、参数列表、回调函数以及服务器响应数据类型。Get 方法会有浏览器缓存问题、Post 方法没有。

2、$.ajax()

这个方法需要通过 js 对象配置请求的相关参数，例如：

```
$.ajax({
    type:请求方式,
    url: 请求地址,
    data:请求参数,
    dataType: 服务器返回值类型,
    success:回调函数
});
```

SpringMVC、Mybatis、Spring

1. *请写出 spring 中常用的依赖注入方式。

常见的就是 setter 注入 和 构造方法 注入。

另外还有静态工厂的方法注入、实例工厂的方法注入。

2. 简述 Spring 中 IOC 容器常用的接口和具体的实现类。

1. BeanFactory SpringIOC容器的基本设置，是最底层的实现， 面向框架本身的.

2. ApplicationContext BeanFactory的子接口，提供了更多高级的特定. 面向开发者的.

3. ConfigurableApplicationContext, ApplicationContext的子接口，扩展出了close 和 refresh等 关闭 刷新容器的方法

4.ClassPathXmlApplicationContext：从 classpath 的 XML 配置文件中读取上下文，并生成上下文定义。应用程序上下文从程序环境变量中取得。

5、FileSystemXmlApplicationContext ：由文件系统中的 XML 配置文件读取上下文。

6、XmlWebApplicationContext：由 Web 应用的 XML 文件读取上下文。

3. 简述 Spring 中如何基于注解配置 Bean 和装配 Bean,

(1). 首先要在 Spring 中配置开启注解扫描

```
<context:component-scan base-package=" " ></context:component-scan>
```

(2). 在具体的类上加上具体的注解.

(3). Spring 中通常使用@Autowired 或者是@Resource 等注解进行 bean 的装配.

4. 说出 Spring 或者 Springmvc 中常用的 5 个注解 ，并解释含义

[1]. @Component 基本注解，标识一个受 Spring 管理的组件

[2]. @Controller 标识为一个表示层的组件

[3]. @Service 标识为一个业务层的组件

[4]. @Repository　　标识为一个持久层的组件

[5]. @Autowired　　自动装配

[6]. @Qualifier(“”)　　具体指定要装配的组件的 id 值

[7]. @RequestMapping()　完成请求映射

[8]. @PathVariable　　映射请求 URL 中占位符到请求处理方法的形参

只要说出 5 个注解并解释含义即可，如上答案只做参考

5. 请解释 Spring Bean 的生命周期？

1.默认情况下，IOC容器中bean的生命周期分为五个阶段:

① 调用构造器 或者是通过工厂的方式创建Bean对象

② 给bean对象的属性注入值

③ 调用初始化方法，进行初始化， 初始化方法是通过init-method来指定的.

④ 使用

⑤ IOC容器关闭时， 销毁Bean对象.

2.当加入了Bean的后置处理器后，IOC容器中bean的生命周期分为七个阶段:

① 调用构造器 或者是通过工厂的方式创建Bean对象

② 给bean对象的属性注入值

③ 执行Bean前置处理器中的

postProcessBeforeInitialization

④ 调用初始化方法，进行初始化， 初始化方法是通过init-method来指定的.

⑤ 执行Bean的后置处理器中 postProcessAfterInitialization

⑥ 使用

⑦ IOC容器关闭时， 销毁Bean对象

只需要回答出第一点即可。 第二点也回答可适当加分.

6. 简述 SpringMvc 里面拦截器是如何定义，如何配置，拦截器中三个重要的方法

(1).定义: 有两种方式

[1]. 实现 HandlerInterceptor 接口

[2]. 继承 HandlerInterceptorAdapter

(2).配置:

```
<mvc:interceptors>
<!--默认是对所有请求都拦截 -->
<bean id="myFirstInterceptor"
class="com.atguigu.interceptor.MyFirstInterceptor">
</bean>
<!-- 只针对部分请求拦截或者不拦截 -->
<mvc:interceptor>
<mvc:mapping path=" " /> <!—指定拦截-->
<mvc:exclude-mapping path=” ” /> <!—指定不拦截-->
```

```
<bean                                                    class="
com.atguigu.interceptor.MySecondInterceptor        "        />
</mvc:interceptor>
</mvc:interceptors>
```

(3).拦截器中三个重要的方法

[1]. preHandle

[2]. postHandle

[3]. afterCompletion

7. 简单的谈一下 SpringMVC 的工作流程？

1、用户发送请求至前端控制器 DispatcherServlet

2、DispatcherServlet 收到请求调用 HandlerMapping 处理器映射器。

3、处理器映射器找到具体的处理器，生成处理器对象及处理器拦截器(如果有则生成)一并返回给 DispatcherServlet。

4、DispatcherServlet 调用 HandlerAdapter 处理器适配器

5、HandlerAdapter 经过适配调用具体的处理器(Controller ,也叫后端控制器)。

6、Controller 执行完成返回 ModelAndView

7、HandlerAdapter 将 controller 执行结果 ModelAndView 返回给 DispatcherServlet

8、DispatcherServlet 将 ModelAndView 传给 ViewReslover 视图解析器

9、ViewReslover 解析后返回具体 View

10、DispatcherServlet 根据 View 进行渲染视图(即将模型数据填充至视图中)。

11、DispatcherServlet 响应用户

8. Springmvc 中如何解决 POST 请求中文乱码问题

Springmvc 中通过 CharacterEncodingFilter 解决中文乱码问题.

在 web.xml 中加入：

```
<filter>
    <filter-name>CharacterEncodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <init-param>
        <param-name>encoding</param-name>
        <param-value>utf-8</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>CharacterEncodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

9. MyBatis 中 #{}和${}的区别是什么？

#{}是预编译处理，${}是字符串替换。

Mybatis 在处理#{}时，会将 sql 中的#{}替换为?号，调用 PreparedStatement 的 set 方法来赋值；

Mybatis 在处理${}时，就是把${}替换成变量的值。

使用#{}可以有效的防止 SQL 注入，提高系统安全性。

10. Mybatis 结果集 的 映射方式有几种，并分别解释每种映射方式如何使用。

自动映射 ，通过 resultType 来指定要映射的类型即可。

自定义映射 通过 resultMap 来完成具体的映射规则，指定将结果集中的哪个列映射到对象的哪个属性。

11.简述 MyBatis 的单个参数、多个参数如何传递及如何取值。

MyBatis 传递单个参数，如果是普通类型(String+8 个基本)的，取值时在#{}中可以任意指定，如果是对象类型的，则在#{}中使用对象的属性名来取值

MyBatis 传递多个参数，默认情况下，MyBatis 会对多个参数进行封装 Map. 取值时

在#{}可以使用 0 1 2 .. 或者是 param1 param2..

MyBatis 传递多个参数，建议使用命名参数，在 Mapper 接口的方法的形参前面使用

@Param() 来指定封装 Map 时用的 key. 取值时在#{}中使用@Param 指定的 key.

12.MyBatis 如何获取自动生成的(主)键值?

在<insert>标签中使用 useGeneratedKeys 和 keyProperty 两个属性来获取自动生成的主键值。

示例:

<insert id=”insertname” usegeneratedkeys=”true” keyproperty=”id”>

insert into names (name) values (#{name})

</insert>

13.简述 Mybatis 的动态 SQL，列出常用的 6 个标签及作用

动态 SQL 是 MyBatis 的强大特性之一 基于功能强大的 OGNL 表达式。

动态 SQL 主要是来解决查询条件不确定的情况，在程序运行期间，根据提交的条件动态的完成查询

常用的标签:

<if>：进行条件的判断

<where>：在<if>判断后的 SQL 语句前面添加 WHERE 关键字，并处理 SQL 语句开始位置的 AND 或者 OR 的问题

<trim>：可以在 SQL 语句前后进行添加指定字符 或者去掉指定字符.

<set>： 主要用于修改操作时出现的逗号问题

<choose> <when> <otherwise> :类似于 java 中的 switch 语句.在所有的条件中选择其一

<foreach>：迭代操作

14.Mybatis 的 Xml 映射文件中，不同的 Xml 映射文件，id 是否可以重复？

不同的 Xml 映射文件，如果配置了 namespace，那么 id 可以重复；如果没有配置 namespace，那么 id 不能重复；

15.Mybatis 如何完成 MySQL 的批量操作,举例说明

MyBatis 完成 MySQL 的批量操作主要是通过<foreache>标签来拼装相应的 SQL 语句.

例如:

```
<insert id="insertBatch" >
    insert into tbl_employee(last_name,email,gender,d_id) values
  <foreach collection="emps" item="curr_emp" separator=",">
(#{curr_emp.lastName},#{curr_emp.email},#{curr_emp.gender},#{curr_emp.dept.id})
  </foreach>
```

</insert>

16. 简述 Spring 中如何给 bean 对象注入集合类型的属性。

Spring 使用 <list> <set> <map> 等标签给对应类型的集合注入值

17.*简述 Spring 中 bean 的作用域

1、singleton 表示在 spring 容器中的单例，通过 spring 容器获得该 bean 时总是返回唯一的实例

2、prototype 表示每次获得 bean 都会生成一个新的对象

3、request 表示在一次 http 请求内有效（只适用于 web 应用）

4、session 表示在一个用户会话内有效（只适用于 web 应用）

5、global session 表示在全局会话内有效（只适用于 web 应用）

18.简述 Spring 中自动装配常用的两种装配模式

byName: 根据 bean 对象的属性名 进行装配

byType：根据 bean 对象的属性的类型进行装配,需要注意匹配到多个兼容类型的 bean 对象时，会抛出异常.

19.简述动态代理的原理， 常用的动态代理的实现方式

动态代理的原理: **使用一个代理将对象包装起来**，然后用该代理对象取代原始对象。任何对原始对象的调用都要通过代理。

代理对象决定是否以及何时将方法调 用转到原始对象上

动态代理的方式

基于接口实现动态代理： JDK 动态代理

基于继承实现动态代理： Cglib、Javassist 动态代理

20.请解释@Autowired 注解的工作机制及 required 属性的作用

1. 首先会使用 byType 的方式进行自动装配，如果能唯一匹配，则装配成功，

   如果匹配到多个兼容类型的 bean，还会尝试使用 byName 的方式进行唯一确定.

   如果能唯一确定，则装配成功，如果不能唯一确定，则装配失败，抛出异常.

2. 默认情况下， 使用@Autowired 标注的属性必须被装配，如果装配不了，也会抛出异常.

   可以使用 required=false 来设置不是必须要被装配.

21.请解释简述 Springmvc 中 ContextLoaderListener 的作用以及实现原理

1. 作用:ContextLoaderListener 的作用是通过监听的方式在 WEB 应用服务器启动时将

Spring 的容器对象进行初始化.

2. 原理: ContextLoaderListener 实现了 ServletContextListener 接口，用于监听 ServletContext 的创建，当监听到 ServletContext 创建时，在对应 contextInitialized 方法中，将 Spring 的容器对象进行创建，并将创建好的容器对象设置到 ServletContext 域对象中，

目的是让各个组件可以通过 ServletContext 共享到 Spring 的容器对象

22.简述 Mybatis 提供的两级缓存，以及缓存的查找顺序

MyBatis 的缓存分为一级缓存和 二级缓存。

一级缓存是 SqlSession 级别的缓存，默认开启。

二级缓存是 NameSpace 级别(Mapper)的缓存，多个 SqlSession 可以共享，使用时需要进行配置开启。

缓存的查找顺序: 二级缓存　　一级缓存　　数据库

23.简述 Spring 与 Springmvc 整合时，如何解决 bean 被创建两次的问题

Bean 被创建两次的问题是在组建扫描的配置中指定 Springmvc 只负责扫描 WEB 相关的组件，Spring 扫描除了 Springmvc 之外的组件。

24.简述 Spring 与 Mybatis 整合时，主要整合的两个地方:

1. SqlSession 创建的问题，通过 SqlSessionFactoryBean 来配置用于创建 SqlSession 的信息。例如: Mybatis 的核心配置文件、Mapper 映射文件、数据源等
2. Mapper 接口创建的问题， 使用 MapperScannerConfigurer 批量为 MyBatis 的 Mapper 接口生成代理实现类并将具体的对象交给 Spring 容器管理

25.简述 Spring 声明式事务中@Transaction 中常用的两种事务传播行为

通过 propagation 来执行事务的传播行为

REQUIRED: 使用调用者的事务，如果调用者没有事务，则启动新的事务运行

REQUIRES_NEW: 将调用者的事务挂起，开启新的事务运行。

26.简述@RequestMapping 注解的作用 可标注的位置 常用的属性

1. 该注解的作用是用来完成请求 与 请求处理方法的映射
2. 该注解可以标注在类上或者是方法上
3. 常用的属性:

value: 默认属性， 用于指定映射的请求 URL

method: 指定映射的请求方式

params: 指定映射的请求参数

headers: 指定映射的请求头信息

27.简述 Springmvc 中处理模型数据的两种方式

1. 使用 ModelAndView 作为方法的返回值，将 模型数据 和 视图信息封装到 ModelAndView 中
2. 使用 Map 或者是 Model 作为方法的形参. 将模型数据添加到 Map 或者是 Model 中.

28.简述 REST 中的四种请求方式及对应的操作

GET 查询操作

POST 添加操作

DELETE 删除操作

PUT 修改操作

29.简述 视图 和 视图解析的关系及作用

1. 视图是由视图解析器解析得到的。
2. 视图解析器的作用是根据 ModelAndView 中的信息解析得到具体的视图对象
3. 视图的作用是完成模型数据的渲染工作，最终完成转发或者是重定向的操作

30.说出三个常用的视图类

InternalResourceView

JstlView

RedirectView

31.简述 REST 中　HiddenHttpMethodFilter 过滤器的作用

该过滤器主要负责转换客户端请求的方式 ,当浏览器的请求方式为 POST,并且在请求中能通过 _method 获取到请求参数值,

该过滤器就会进行请求方式的转换.

一般在 REST 中，都是将 POST 请求转换为对应的 DELETE 或者是 PUT

32.简述 Springmvc 中如何返回 JSON 数据

1. 在工程最终加入 jackson 的 jar 包
2. 在请求处理方法中 将返回值改为具体返回的数据的类型， 例如 数据的集合类型 List<Employee>等
3. 在请求处理方法上使用@ResponseBody 注解

33.简述如何在 myBatis 中的增删改操作获取到对数据库的影响条数

直接在 Mapper 接口的方法中声明返回值即可.

34.Springmvc 中的控制器的注解用哪个，可以是否用别的注解代替

使用@Controller 注解来标注控制器。 不能使用别的注解代替

35.如何在 Springmvc 中获取客户端提交的请求参数

直接在请求处理方法中声明对应的形参，也可以是用@RequestParam 注解来具体指定将那些请求参数映射到方法中对应的形参。

36.简述 Springmvc 中 InternalResourceViewResolver 解析器的工作机制

使用 prefix + 方法的返回值 + suffix 生成一个物理视图路径。

37.Springmvc 中如何完成重定向

在请求处理方法的返回值前面加 redirect: 前缀，最终会解析得到 RedirectView，RedirectView 会完成重定向的操作.

38.简述 Spring 中切面中常用的几种通知，并简单解释

前置通知 在目标方法执行之前执行

后置通知 在目标方法执行之后执行，不管目标方法有没有抛出异常

返回通知 在目标方法成功返回之后执行， 可以获取到目标方法的返回值.

异常通知 在目标方法抛出异常后执行

环绕通知 环绕着目标方法执行，

39.解释 MyBatis 中 @Param 注解的作用

通过该注解来指定 Mybatis 底层在处理参数时封装 Map 使用的 key，方便在 SQL 映射文件中取参数.

40.简述 Mybatis 中使用 Mapper 接口开发，如何完成 Mapper 接口与 SQL 映射文件、方法与 SQL 语句的绑定

Mapper 接口与 SQL 映射文件绑定: SQL 映射文件中的 namespace 的值指定成 Mapper 接口的全类名

接口中方法与 SQL 语句的绑定: SQL 语句的 id 指定成接口中的方法名.

41.SpringMVC 的工作原理

（1）用户向服务器发送请求，请求被 springMVC 前端控制器 DispatchServlet 捕获；

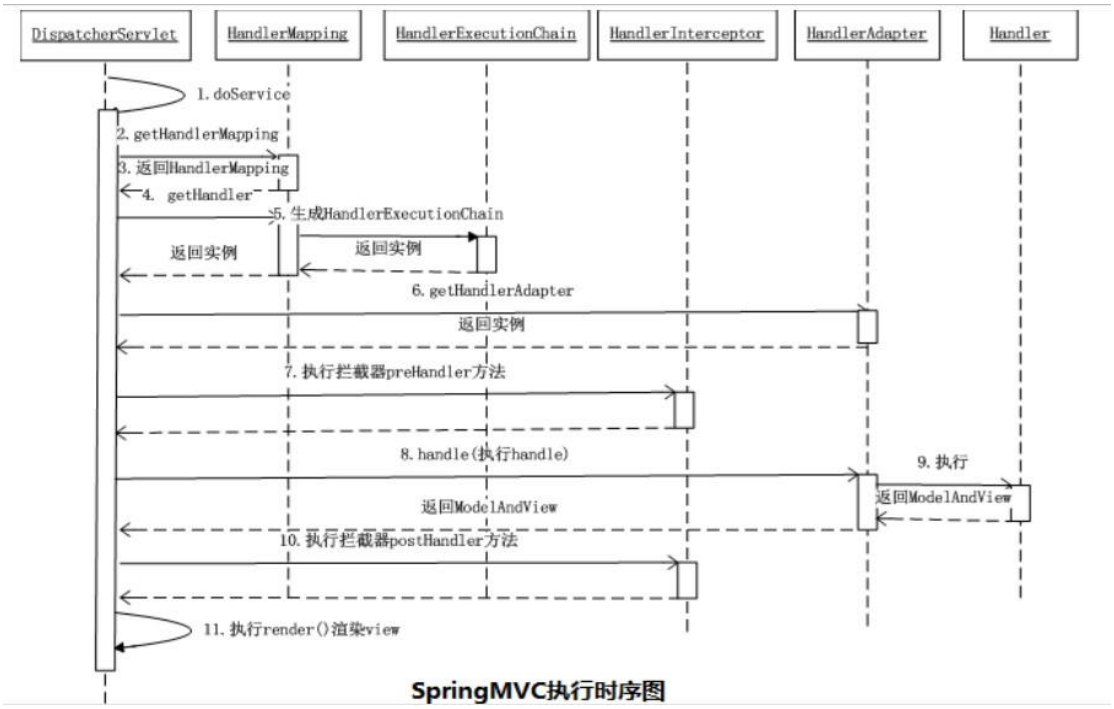
（2）DispatcherServle 对请求 URL 进行解析，得到请求资源标识符（URL），然后根据该 URL 调用 HandlerMapping 将请求映射到处理器 HandlerExcutionChain；

（3）DispatchServlet 根据获得 Handler 选择一个合适的 HandlerAdapter 适配器处理；

（4）Handler 对数据处理完成以后将返回一个 ModelAndView（）对象给 DisPatchServlet;

（5）Handler 返回的 ModelAndView() 只是一个逻辑视图并不是一个正式的视图，DispatcherSevlet 通过 ViewResolver 试图解析器将逻辑视图转化为真正的视图 View;

（6）DispatcherServle 通过 model 解析出 ModelAndView()中的参数进行解析最终展现出完整的 view 并返回给客户端;



SpringMVC执行时序图

42.谈谈你对 Spring 的理解

Spring 是一个开源框架，为简化企业级应用开发而生。Spring 可以是使简单的 JavaBean 实现以前只有 EJB 才能实现的功能。Spring 是一个 IOC 和 AOP 容器框架。

Spring 容器的主要核心是：

控制反转(IOC),传统的 java 开发模式中,当需要一个对象时,我们会自己使用 new 或者 getInstance 等直接或者间接调用构造方法创建一个对象。而在 spring 开发模式中，spring 容器使用了工厂模式为我们创建了所需要的对象，不需要我们自己创建了，直接调用 spring 提供的对象就可以了，这是控制反转的思想。

依赖注入(DI),spring 使用 javaBean 对象的 set 方法或者带参数的构造方法为我们在创建所需对象时将其属性自动设置所需要的值的过程，就是依赖注入的思想。

面向切面编程(AOP),在面向对象编程(oop)思想中，我们将事物纵向抽成一个个的对象。而在面向切面编程中,我们将一个个的对象某些类似的方面横向抽成一个切面,对这个切面进行一些如权限控制、事物管理,记录日志等公用操作处理的过程就是面向切面编程的

思想。AOP 底层是动态代理，如果是接口采用 JDK 动态代理，如果是类采用 CGLIB 方式实现动态代理。

43.Spring 中常用的设计模式

（1） 代理模式——spring 中两种代理方式，若目标对象实现了若干接口，spring 使用 jdk 的 java.lang.reflect.Proxy 类代理。若目标兑现没有实现任何接口，spring 使用 CGLIB 库生成目标类的子类。

（2） 单例模式——在 spring 的配置文件中设置 bean 默认为单例模式。

（3） 模板方式模式——用来解决代码重复的问题。

比如：RestTemplate、JmsTemplate、JpaTemplate

（4） 工厂模式——在工厂模式中，我们在创建对象时不会对客户端暴露创建逻辑，并且是通过使用同一个接口来指向新创建的对象。Spring 中使用 beanFactory 来创建对象的实例。

44.请描述一下 Spring 的事务管理

1、声明式事务管理的定义：用在 Spring 配置文件中声明式的处理事务来代替代码式的处理事务。这样的好处是，事务管理不侵入开发的组件，具体来说，业务逻辑对象就不会意识到正在事务管理之中，事实上也应该如此，因为事务管理是属于系统层面的服务，而不是业务逻辑的一部分，如果想要改变事务管理策划的话，也只需要在定义文件中重新配置即可，这样维护起来极其方便。

基于 TransactionInterceptor 的声明式事务管理：两个次要的属性：transactionManager，用来指定一个事务治理器， 并将具体事务相关的操作请托给它；其他一个是 Properties 类型的 transactionAttributes 属性，该属性的每一个键值对中，键指定的是方法名，方法名可以行使通配符， 而值就是表现呼应方法的所运用的事务属性。

2、基于 @Transactional 的声明式事务管理：Spring 2.x 还引入了基于 Annotation 的体式格式，具体次要触及@Transactional 标注。@Transactional 可以浸染于接口、接口方法、类和类方法上。算作用于类上时，该类的一切 public 方法将都具有该类型的事务属性。

3、编程式事物管理的定义 :在代码中显式挪用 beginTransaction()、commit()、rollback()等事务治理相关的方法， 这就是编程式事务管理。Spring 对事物的编程式管理有基于底层 API 的编程式管理和基于 TransactionTemplate 的编程式事务管理两种方式。

45.谈谈 AOP

AOP，Aspect Oriented Programming 面向切面编程。可以使开发人员在不修改原来的代码的基础上，增加系统的新功能。

AOP 主要应用于日志记录，性能统计，安全控制，事务处理（项目中使用的）等方面。

通过预编译方式和运行期动态代理实现程序功能的统一维护的一种技术。

46.谈谈 IOC 和 DI

IOC 是 Inverse Of Control 的简写，意思是控制反转。是降低对象之间的耦合关系的设计思想。

通过 IOC，开发人员不需要关心对象的创建过程，交给 Spring 容器完成。具体的过程是，程序读取 Spring 配置文件，获取需要创建的 bean 对象，通过反射机制创建对象的实例，并通过 DI 注入对象的依赖关系。

DI 是 Dependency Injection 依赖注入，说的是创建对象实例时，同时为这个对象注入它所依赖的属性。相当于把每个 bean 与 bean 之间的关系交给容器管理。而这个容器就是 spring。例如我们通常在 Service 层 注入 它所依赖的 Dao 层的实例；在 Controller 层注入 Service 层的实例。

47.Spring 的底层实现机制是什么？

使用 Dom4j（解析 XML）+ Java 反射机制

Dom4j 其实就是解析 XML。使用反射机制实例化 bean。

48.Spring 配置文件 applicationContext.xml 可以修改名称吗？

可以修改，在 web.xml 文件中配置，如下：

```
<context-param>
```

```
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:bean.xml</param-value>
  </context-param>
```

49.Spring 配置文件中都有哪些配置

数据源 dataSource，sessionFactoryBean，service 层和 dao 层的注解扫描，事物，其它中间件对象的初始化例如 jedis

50.Spring 容器的 bean 是单例的吗

默认是单例的。但是如果配置了 scope = “prototype” ，则是多实例的。scope 的 默认 值 是 "singleton"，表示单实例。

51.Spring MVC 的 Controller 是单例的吗？是线程安全的吗？

是单例的。是线程安全的。

原因：Spring MVC 虽然是单例的， 但是 Spring MVC 是方法级别的，数据传输是通过方法的参数传递的，因此在类中没有共享属性，不存在线程不安全的问题。

52.Spring 和 springMVC 的区别？

Spring 是一个 Java EE 框架，支持 IOC、DI 和 AOP。Spring 框架有很多模块组成，每个模块可以独立运行。

Spring MVC 是 Spring 的一个模块，实现 MVC 设计思想。

53.SpringMVC 中都有哪些配置

视图解析器、放行静态资源、扫描 controller、文件上传下载、json 类型转换器、自定义拦截器、全局异常处理

54.Spring，SpringMVC ，mybatis 分别是解决什么问题的？

SpringMVC 是 MVC 框架，主要是将系统分为视图、模型、控制器三个层次，负责接收请求，调用业务层接口，给视图层返回需要的数据。降低系统的耦合性。

mybatis 是持久层框架，解决如何存取数据库中的数据的问题，提高数据操作的效率。

Spring 实现依赖注入的功能，例如将 dao 对象注入 service 层，同时 spring 面向切面的特性可以实现程序中事物管理的功能，日志记录的功能等。

55.MyBatis 的三层架构

1、基础支撑层：负责读取 mybatis 配置文件，链接数据库，做事物管理

2、数据处理层：解析 sql、执行 sql、映射结果集

3、API 接口层：提供操作数据的 api

56.写出 MyBatis 配置自定义映射使用的标签，并解释含义

1) 自定义 resultMap，实现高级结果集映射

2) id ：用于完成主键值的映射

3) result ：用于完成普通列的映射

4) association ：一个复杂的类型关联;许多结果将包成这种类型

5) collection ： 复杂类型的集

57.写出 MyBatis 一级缓存失效的几种情况

1) 不同的 SqlSession 对应不同的一级缓存

2) 同一个 SqlSession 但是查询条件不同

3) 同一个 SqlSession 两次查询期间执行了任何一次增删改操作

4) 同一个 SqlSession 两次查询期间手动清空了缓存

Java 高级

1、 vi/vim 三种模式是什么？三种模式如何切换？

2、 vi/vim 三种模式：一般模式（默认模式）、编辑模式、命令模式

3、 切换：

4、（1）vi/vim 文件进入一般模式

5、（2）一般模式下按 i、a、o 进入编辑模式，esc 退出

6、（3）一般模式下，按：、/进入命令模式，esc 退出

7、（4）命令模式下:q! 不保存退出，:wq!保存退出

2、如何查看 linux 系统防火墙（iptables）服务是否启动？如未启动，如何启动？

（1）运行命令：service iptables status

（2）运行命令：service iptables start

3、Linux 常见打包工具并写相应解压缩参数?

①用 tar 调用 bzip2 将 file1,file2,file3 生成压缩包

tar -jcvf file.tar.bz2 file{1,2,3} 解压：tar -jxvf file.tar.bz2

②用 tar 调用 gzip 将 file1,file2,file3 生成压缩包

tar -zcvf file.tar.gz file{1,2,3} 解压：tar -zxvf file.tar.gz

③.用 tar 调用 xz 将 file1,file2,file3 生成压缩包

tar -Jcvf file.tar.xz file{1,2,3} 解压：tar -Jxvf file.tar.xz

4、什么是符号链接，什么是硬链接？符号链接与硬链接的区别是什么？

答：链接分硬链接和符号链接。符号链接可以建立对于文件和目录的链接。符号链接可以跨磁盘分区，符号链接的文件类型位是 l，链接文件具有新的 i 节点；硬链接不可以跨磁盘分区。它只能建立对文件的链接，硬链接的文件类型位是-号，且硬链接文件的 i 节点同被链接文件的 i 节点相同。

5、5、git 中 reset 与 rebase, pull 与 fetch 的区别

git reset 不修改 commit 相关的东西，只会去修改.git 目录下的东西。

git rebase 会试图修改你已经 commit 的东西，比如覆盖 commit 的历史等，但是不能使用 rebase 来修改已经 push 过的内容，容易出现兼容性问题。rebase 还可以来解决内容的冲突，解决两个人修改了同一份内容，然后失败的问题。

git pull pull=fetch+merge,

使用 git fetch 是取回远端更新，不会对本地执行 merge 操作，不会去动你的本地的内容。

pull 会更新你本地代码到服务器上对应分支的最新版本

6、git 如何解决代码冲突

git stash

git pull

git stash pop

这个操作就是把自己修改的代码隐藏，然后把远程仓库的代码拉下来，然后把自己隐藏的修改的代码释放出来，让 gie 自动合并。

如果要代码库的文件完全覆盖本地版本。

git reset –hard

git pull

7、Web Service 的基本原理

(1) Service Provider 采用 WSDL 描述服务

(2) Service Provider 采用 UDDI 将服务的描述文件发布到 UDDI 服务器（Register server）

(3) Service Requestor 在 UDDI 服务器上查询并 获取 WSDL 文件

(4) Service requestor 将请求绑定到 SOAP，并访问相应的服务。

8、如何发布一个 webservice

1.定义 SEI（接口） @webservice（类） @webMethod（暴露的方法）

2.定义 SEI 的实现

3.发布 Endpoint.publish(url,new SEI 的实现对象)

9、Redis 是单线程还是多线程？特点都有那些?

（1）redis 是单线程的

（2）数据都存在内存中

（3）支持持久化，主要用作备份恢复

（4）key-value 支持多种数据结构存储：string、list、set、hash、zset

（5）redis 支持集群，数据分布式存储

（6）因受物理内存限制，不能做海量数据的高性能读写，适用在较小数据量的高性能操作和运算上

10、如何实现 redis 的事务？如果事务中的一个命令写错了，整体是否成功？如果事务中一个命令执行时报错，整体是否成功？

（1）multi 开始事务，exec 执行事务

（2）命令写错，整体失败

（3） 执行报错，除报错命令外其他的都执行

11、RDB 与 AOF 各自的优缺点？如同时开启系统默认取谁的数据？

（1）rdb 优点：节省磁盘空间、恢复速度快；

缺点：比较消耗性能、两次备份间 redis down 掉了，会丢失最后一次备份后的修改

aof 优点：备份机制更稳健，丢失数据概率更低、可读的日志文本、可以处理误操作；

缺点：比 rdb 占更多磁盘空间、恢复速度慢、每次读写同步有性能压力、存在 bug 会造成不能恢复

（2） 两个同时开启，系统默认取 aof 的数据

12、一台主机 master，两台从机(slave1,slave2)，如主机 shutdown 了，主从关系是否会变化？什么情况下变化？

（1）不会变化

（2）配置哨兵模式后变化，哨兵会把一台从机切换成主机

13、Redis 支持的数据类型

Redis 通过 Key-Value 的单值不同类型来区分,

以下是支持的类型:Strings

Lists

Sets 求交集、并集

Sorted Set

hashes

14、redis 常见性能问题和解决方案

1).Master 写内存快照，save 命令调度 rdbSave 函数，会阻塞主线程的工作，当快照比较大时对性能影响是非常大的，会间断性暂停服务，所以 Master 最好不要写内存快照。

2).Master AOF 持久化，如果不重写 AOF 文件，这个持久化方式对性能的影响是最小的，但是 AOF 文件会不断增大，AOF 文件过大会影响 Master 重启的恢复速度。Master 最好不要做任何持久化工作 ,包括内存快照和 AOF 日志文件 ,特别是不要启用内存快照做持久化,如果数据比较关键 ,某个 Slave 开启 AOF 备份数据，策略为每秒同步一次。

3).Master 调用 BGREWRITEAOF 重写 AOF 文件，AOF 在重写的时候会占大量的 CPU 和内存资源，导致服务 load 过高，出现短暂服务暂停现象。

4). Redis 主从复制的性能问题 ,为了主从复制的速度和连接的稳定性 ,Slave 和 Master 最好在同一个局域网内

15、为什么要用 Nginx？--11

优点：

跨平台、配置简单

非阻塞、高并发连接：处理 2-3 万并发连接数，官方监测能支持 5 万并发

内存消耗小：开启 10 个 nginx 才占 150M 内存，Nginx 采取了分阶段资源分配技术

Nginx 处理静态文件好,耗费内存少

内置的健康检查功能：如果有一个服务器宕机，会做一个健康检查，再发送的请求就不会发送到宕机的服务器了。重新将请求提交到其他的节点上。

节省宽带：支持 GZIP 压缩，可以添加浏览器本地缓存

稳定性高：宕机的概率非常小

master/worker 结构：一个 master 进程，生成一个或者多个 worker 进程

接收用户请求是异步的：浏览器将请求发送到 nginx 服务器，它先将用户请求全部接收下来，再一次性发送给后端 web 服务器，极大减轻了 web 服务器的压力

一边接收 web 服务器的返回数据，一边发送给浏览器客户端

网络依赖性比较低，只要 ping 通就可以负载均衡

可以有多台 nginx 服务器

事件驱动：通信机制采用 epoll 模型

16、Nginx 反向代理

反向代理（Reverse Proxy）方式是指以代理服务器来接受 internet 上的连接请求，然后将请求，发给内部网络上的服务器

并将从服务器上得到的结果返回给 internet 上请求连接的客户端，此时代理服务器对外就表现为一个反向代理服务器

反向代理总结就一句话：代理端代理的是服务端

17、动态资源，静态资源分离？为什么要做动、静分离？--12

1)、动态资源、静态资源分离是让动态网站里的动态网页根据一定规则把不变的资源和经常变的资源区分开来，动静资源做好了拆分以后，我们就可以根据静态资源的特点将其做缓存操作，这就是网站静态化处理的核心思路

动态资源、静态资源分离简单的概括是：动态文件与静态文件的分离

在我们的软件开发中，有些请求是需要后台处理的（如：.jsp,.do 等等），有些请求是不需要经过后台处理的（如：css、html、jpg、js 等等文件）

2)、这些不需要经过后台处理的文件称为静态文件，否则动态文件。因此我们后台处理忽略静态文件。这会有人又说那我后台忽略静态文件不就完了吗，当然这是可以的，但是这样后台的请求次数就明显增多了。在我们对资源的响应速度有要求的时候，我们应该使用这种动静分离的策略去解决动、静分离将网站静

态资源（HTML，JavaScript，CSS，img 等文件）与后台应用分开部署，提高用户访问静态代码的速度，降低对后台应用访问

这里我们将静态资源放到 nginx 中，动态资源转发到 tomcat 服务器中

18、Nginx 负载均衡--13

负载均衡即是代理服务器将接收的请求均衡的分发到各服务器中

负载均衡主要解决网络拥塞问题，提高服务器响应速度，服务就近提供，达到更好的访问质量，减少后台服务器大并发压力。

19、举出 4 种常用的 log 级别，并按照输出内容由少到多依次排列

rror,warn,info,debug

20、简述 MySQL 数据引擎 myisam 与 innodb 的区别

(1)、问 5 点不同；

1>.InnoDB 支持事物，而 MyISAM 不支持事物

2>.InnoDB 支持行级锁，而 MyISAM 支持表级锁

3>.InnoDB 支持 MVCC, 而 MyISAM 不支持

4>.InnoDB 支持外键，而 MyISAM 不支持

5>.InnoDB 不支持全文索引，而 MyISAM 支持。

(2)、innodb 引擎的 4 大特性

插入缓冲（ insert buffer),二次写(double write),自适应哈希索引(ahi),预读(read ahead)

(3)、2 者 selectcount(*)哪个更快，为什么

myisam 更快，因为 myisam 内部维护了一个计数器，可以直接调取。

21、索引概述

1、索引的概念

索引就是加快查询表中数据的方法。

数据库的索引类似于书籍的索引。

在书籍中，索引允许用户不必翻阅完整个书就能迅速地找到所需要的信息。

在数据库中，索引也允许数据库程序迅速地找到表中的数据，

而不必扫描整个数据库.

2、索引的特点

1.索引可以加快数据库的检索速度

2.索引降低了数据库插入、修改、删除等维护任务的速度

3.索引创建在表上，不能创建在视图上

3、索引的优点

1.创建唯一性索引，保证数据库表中每一行数据的唯一性

2.大大加快数据的检索速度，这也是创建索引的最主要的原因

3.减少磁盘 IO（向字典一样可以直接定位）

4、索引的缺点

1.创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加

2.索引需要占物理空间，除了数据表占数据空间之外，每一个索引还要占一定 的物理空间

3.当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，降低 了数据的维护速度

5、索引的分类

1.普通索引和唯一性索引

普通索引：CREATE INDEX mycolumn_index ON mytable (myclumn)

唯一性索引：保证在索引列中的全部数据是唯一的

CREATE unique INDEX mycolumn_index ON mytable (myclumn)

2. 单个索引和复合索引

单个索引：即非复合索引

复合索引：又叫组合索引，在索引建立语句中同时包含多个字段名，最多 16 个字段

CREATE INDEX name_index ON username(firstname,lastname)

顺序索引，散列索引,位图索引

22、innodb 的事务与日志的实现方式

(1)、有多少种日志；

错误日志：记录出错信息，也记录一些警告信息或者正确的信息。

查询日志 :记录所有对数据库请求的信息 ,不论这些请求是否得到了正确的执行。

慢查询日志：设置一个阈值，将运行时间超过该值的所有 SQL 语句都记录到慢查询的日志文件中。

二进制日志：记录对数据库执行更改的所有操作。

中继日志：

事务日志：

(2)、事物的 4 种隔离级别

隔离级别

读未提交(RU)

读已提交(RC)

可重复读(RR)

串行

(3)、事务是如何通过日志来实现的，说得越深入越好。

事务日志是通过 redo 和 innodb 的存储引擎日志缓冲（Innodb log buffer）来实现的，当开始一个事务的时候，会记录该事务的 lsn(log sequence number)号; 当事务执行时，会往 InnoDB 存储引擎的日志

的日志缓存里面插入事务日志；当事务提交时，必须将存储引擎的日志缓冲写入磁盘（通过 innodb_flush_log_at_trx_commit 来控制），也就是写数据前，需要先写日志。这种方式称为“预写日志方式”

23、MySQL 数据库 cpu 飙升到 500%的话他怎么处理？

(1)、没有经验的，可以不问；

(2)、有经验的，问他们的处理思路。

列出所有进程　show processlist　观察所有进程　多秒没有状态变化的(干掉)

查看超时日志或者错误日志（做了几年开发,一般会是查询以及大批量的插入会导致 cpu 与 i/o 上涨,,,当然不排除网络状态突然断了,,导致一个请求服务器只接受到一半，比如 where 子句或分页子句没有发送,,当然的一次被坑经历)

24、存储过程概述

存储过程（Stored Procedure）

可以包含逻辑判断的 sql 语句集合。

是经过预编译，存在于数据库中。

通过调用指定存储过程的名字（可有参，可无参）来执行。

优点：

简化了复杂的业务逻辑，根据需要可重复使用

屏蔽了底层细节，不暴露表信息即可完成操作

降低网络的通信量，多条语句可以封装成一个存储过程来执行

设置访问权限来提高安全性

提高执行效率，因为它是预编译以及存储在数据库中

缺点：

可移植性差，相同的存储过程并不能跨多个数据库进行操作

大量使用存储过程后，首先会使服务器压力增大，而且维护难度逐渐增加

存储过程的语法：

--下面是在 oracle 数据库下最基本的语法

--仅创建一个名为 testProcedure 的无参的存储过程

--IS 也可以是 AS

--如果已经存在名为 testProcedure 的存储过程，下面的语法会出现 名称已被使用的错误

--解决办法：

--第一句可以写成 create or replace procedure testProcedure

--这样会替换原有的存储过程

--NULL 表示任何可以正确执行的 sql 语句，但至少一句

create procedure testProcedure IS BEGIN NULL END;

存储过程的参数的分类:

IN

OUT

INOUT

注意：

存储过程之间可相互调用

存储过程一般修改后，立即生效。/

25、jvm 的内存结构

堆： 逻辑上是连续,物理上可以处于不连续的内存空间中，

里面存储的是对象实例以及数组。可以细分为新生代，老生代。

通过-Xmx 和-Xms 控制大小。

虚拟机栈：基本数据类型，对象引用(地址，指针)。

本地方法栈（了解）:它与虚拟机栈发挥的作用差不多，区别在于虚拟机栈为 java 方法的执行提供服务，而本地方法栈为虚拟机使用到的 Native(本地)方法服务。

方法区：放了所加载的类的信息（名称、修饰符等）、类中的静态变量、

类中定义为 final 类型的常量、类中的 Field 信息、类中的方法信息

在 Sun JDK 中这块区域对应的为 PermanetGeneration，又称为持久代，

默认为 64M，可通过-XX:PermSize 以及-XX:MaxPermSize 来指定其大小

在服务器启动的时候报内存溢出是因为方法区太小，也就相当于持久代的内存太小。

通过-XX:PermSize 以及-XX:MaxPermSize 来指定其大小，可以解决这个问题。

常量池是方法区的一部分,用来存储常量信息。如 String 就存储在常量池中。

计数器（了解）:通过该计数器的值来选取下一条要执行的字节码指令。

26、存泄露和内存溢出

内存泄露（memory leak)，是指应用程序在申请内存后，无法释放已经申请的内存空间.一次内存泄露危害可以忽略，但如果任其发展最终会导致内存溢出(out of memory).如读取文件后流要进行及时的关闭以及对数据库连接的释放。

内存溢出（out of memory）是指应用程序在申请内存时，没有足够的内存空间供其使用。如我们在项目中对于大批量数据的导入 ,采用分段批量提交的方式。

27、类的实例化顺序，比如父类静态数据，构造函数，字段，子类静态数据，构造函数，字段，他们的执行顺序

答：先静态、先父后子。

先静态：父静态 > 子静态

优先级：父类 > 子类 静态代码块 > 非静态代码块 > 构造函数

一个类的实例化过程：

1，父类中的 static 代码块，当前类的 static

2，顺序执行父类的普通代码块

3，父类的构造函数

4，子类普通代码块

5，子类（当前类）的构造函数，按顺序执行。

6，子类方法的执行，

28、JVM 垃圾回收机制，何时触发 MinorGC 等操作

分代垃圾回收机制：不同的对象生命周期不同。把不同生命周期的对象放在不同代上，不同代上采用最合适它的垃圾回收方式进行回收。

JVM 中共划分为三个代：年轻代、年老代和持久代，

年轻代：存放所有新生成的对象；

年老代：在年轻代中经历了 N 次垃圾回收仍然存活的对象，将被放到年老代中，故都是一些生命周期较长的对象；

持久代：用于存放静态文件，如 Java 类、方法等。

新生代的垃圾收集器命名为“minor gc”，老生代的 GC 命名为”Full Gc 或者 Major GC”.其中用 System.gc()强制执行的是 Full Gc.

判断对象是否需要回收的方法有两种：

1.引用计数

当某对象的引用数为 0 时，便可以进行垃圾收集。

2.对象引用遍历

果某对象不能从这些根对象的一个（至少一个）到达，则将它作为垃圾收集。在对象遍历阶段，gc 必须记住哪些对象可以到达，以便删除不可到达的对象，这称为标记（marking）对象。

触发 GC（Garbage Collector）的条件：

1)GC 在优先级最低的线程中运行，一般在应用程序空闲即没有应用线程在运行时被调用。

2)Java 堆内存不足时，GC 会被调用。

29、深入分析了 Classloader，双亲委派机制

ClassLoader：类加载器（class loader）用来加载 Java 类到 Java 虚拟机中。Java 源程序(.java 文件)在经过 Java 编译器编译之后就被转换成 Java 字节代码（.class 文件）。类加载器负责读取 Java 字节代码，并转换成 java.lang.Class 类的一个实例。

双亲委派机制：某个特定的类加载器在接到加载类的请求时，首先将加载任务委托给父类加载器，依次递归，如果父类加载器可以完成类加载任务，就成功返回；只有父类加载器无法完成此加载任务时，才自己去加载。/

30、简述 synchronized 和 java.util.concurrent.locks.Lock 的异同？

主要相同点：Lock 能完成 synchronized 所实现的所有功能

主要不同点：Lock 有比 synchronized 更精确的线程语义和更好的性能。synchronized 会自动释放锁，而 Lock 一定要求程序员手工释放，并且必须在 finally 从句中释放。

31、Tomcat 优化

增大内存(堆，持久代)并开启 server 模式

我在做 XXX 项目时,用到了 poi 导入和导出数据,由于公司的业务比较繁多,数据量很大,测试时报内存溢出,经过我的分析再结合上网查阅资料,发现可能是 tomcat 内存不足,需要增大,修改配置文件后测试不再报错.

tomcat 增大内存的方式通过修改 tomcat 配置文件

window 下， 在 bin/catalina.bat 文件中最前面添加：

set JAVA_OPTS=-XX:PermSize=64M -XX:MaxPermSize=128m –Xms1024m -Xmx1024m

linux 下，在 catalina.sh 最前面增加：

JAVA_OPTS="-XX:PermSize=64M -XX:MaxPermSize=128m –Xms1024m -Xmx1024m "

-client –service

当我们在 cmd 中运行-java 时,黑窗口会出现-client -service 这两参数.其作用是设置虚拟机运行模式;client 模式启动比较快 ,但运行时性能和内存管理效率不如 server 模式，通常用于客户端应用程序。server 模式启动比 client 慢，但可获得更高的运行性能。Windows 默认为 client，如果要使用 server 模式，就需要在启动虚拟机时加-server 参数，以获得更高性能，对服务器端应用，推荐采用 server 模式 ,尤其是多个 CPU 的系统。在 Linux ,Solaris 上,默认值为 server 模式.

JDK 版本

影响虚拟机还有 JDK 的版本,JDK 分为 32 位,64 位两种版本,32 位装在 32 位系统,64 位系统可以装 32 位和 64 位 JDK.64 位 JDK 性能优于 32 位 JDK.

测试的命令 java -xmx 数值 m –version 报错配置大小失败,反之成功

增加 Tomcat 最大连接数

使用场景

我在做完一个 XXX 项目后,测试时发现并发数量增加到一定程度就会很卡,于是我想到了是不是 tomcat 最大连接数设置有限制.果不其然,配置文件中最大值才 500,于是我更改了最大连接数,根据业务我修改了连接数为 2000,完美的解决了这个问题;

修改方法在 conf/service.xml 中默认值

<Connector port="8080" maxHttpHeaderSize="8192"
maxThreads="1500"
minSpareThreads="30" maxSpareThreads="75"
enableLookups="false"
redirectPort="8443" acceptCount="100"
connectionTimeout="20000"
disableUploadTimeout="true" />,修改 maxthreads 的值即可

tomcat 进行 gzip 压缩从而降低网络传输量

tomcat 压缩设置 tomcat 压缩 gzip 启用

HTTP 压缩可以大大提高浏览网站的速度，它的原理是，
在客户端请求服务器对应资源后，从服务器端将资源文件压缩，
再输出到客户端，由客户端的浏览器负责解压缩并浏览。
相对于普通的浏览过程 HTML ,CSS,Javascript , Text ，
它可以节省 60%左右的流量。更为重要的是，它可以对动态生成的，
包括 CGI、PHP , JSP , ASP , Servlet,SHTML 等输出的网页也能进行压缩，
压缩效率也很高。

启用 tomcat 的 gzip 压缩

要使用 gzip 压缩功能，你需要在 Connector 节点中加上如下属性

记住来源：http://www.qi788.com/info-42.html

compression="on" 打开压缩功能

compressionMinSize="50" 启用压缩的输出内容大小，默认为 2KB

noCompressionUserAgents="gozilla, traviata" 对于以下的浏览器，不启用压缩

compressableMimeType="text/html,text/xml,text/javascript,text/css,text/plain" 哪些资源类型需要压缩

<Connector port="80" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="8443" executor="tomcatThreadPool" URIEncoding="utf-8"

compression="on"

compressionMinSize="50"

noCompressionUserAgents="gozilla, traviata"

compressableMimeType="text/html,text/xml,text/javascript,text/css,text/plain" />/

32、Zookeeper 对节点的 watch 监听通知是永久的吗？

不是。官方声明：一个 Watch 事件是一个一次性的触发器，当被设置了 Watch 的数据发生了改变的时候 ,则服务器将这个改变发送给设置了 Watch 的客户端，以便通知它们。

为什么不是永久的，举个例子，如果服务端变动频繁，而监听的客户端很多情况下，每次变动都要通知到所有的客户端，这太消耗性能了。

一般是客户端执行 getData(“/节点 A”,true)，如果节点 A 发生了变更或删除，客户端会得到它的 watch 事件，但是在之后节点 A 又发生了变更，而客户端又没有设置 watch 事件，就不再给客户端发送。

在实际应用中，很多情况下，我们的客户端不需要知道服务端的每一次变动，我只要最新的数据即可。/

33、Zookeeper 的通知机制

client 端会对某个 znode 建立一个 watcher 事件，当该 znode 发生变化时，这些 client 会收到 zk 的通知，然后 client 可以根据 znode 变化来做出业务上的改变等。

默认端口号

mycat 8066 ElasticSearch：9200 kibana ：5601
mysql 3306 Tomcat 8080
redis 6379
zookeeper ：2181
ActiveMQ 两个重要的端口，ActiveMQ 两个重要的端口，一个是提供消息队列的默认端口：61616 另个是控制台端口 8161